State-of-the-Art Computer Architecture

CHAPTER 10

CHAPTER OBJECTIVES

- Give an overview on state-of-the-art computer architectures
- Contrast the differences of RISC and CISC
- Classify multiprocessor and multicomputer architectures
- Explain pipelining
- Recent developments in multi-core systems
- Provide an understanding for advanced bus systems
- Give an outlook in latest storage technologies

In our daily life we use many different types of computers. We hear music with an MP3-player while jogging, and your smart phone is a multimedia system that lets you not only phone your friends, listen to the radio but also view a video clip. Refrigerator and dish washer have embedded computers to control the temperature. In your car you could find a dozen of computers, not only the navigation system, car phone, radio, or CD player, but a lot of more computers you are not aware of, except when one of them is not working, and in rare cases the malfunction is only noticed when it is indicated on the dashboard. All these computers are not general purpose computers as your laptop, but are specialised in terms of hardware architecture and operating system.

So, the slogan "one size fits all" does not apply. This is why we will try to give an overview of all types of digital computers by classifying them according to different characteristics. We will present state-of-the-art technologies for miniaturizing storage devices and for boosting processor performance. Nanotechnologies provide the highest possible storage density and miniaturizing the physical dimension of devices. Parallelization of the work load and the assembly line like processing boosts performance without the need to speed up a single operation or device. This is a very important fact as faster devices have physical and electrical limits and consume overproportional power, i.e. produce too much heat which prevents miniaturization due to thermal problems.

10.1 Overview and Classification of Computer Architectures

In the last chapter we have looked at the basic computer architecture as proposed by John von Neumann. After fifty years of computer design experience many different architectures have evolved. Nevertheless from a programmer's point of view the principle mechanism of a von Neumann computer has not changed. On the abstraction layer of the programmer there still exist the sequential instruction processing and the dualism of data and instructions. On the hardware side, the component's technology evolved substantially and the components grew to a sophisticated architecture. Some computers are optimized for specific tasks, like graphics adapters for rendering three dimensional graphics. These manifold of computer architectures rise the need to bring some order into the diversity.

There are many ways to group computer systems, e.g. by

- usage/purpose
- processing power
- power consumption
- technology
- hardware architecture

Classifying a computer system by its usage or purpose is useful but not a sufficiently sharp criterion. There are special purposes for computers that require dedicated hardware and software. Mobile phones are build to consume a minimum of power because independent operation time and weight are essential. Therefore special low power components have been designed and built into the device. Power reduction circuits in MOS-FET (Metal-Oxide Semiconductor Field Effect Transistor) technology in conjunction with software that supports standby and hibernation modes are used for reduced power consumption. In *Standby* mode all functions of the system are halted except the sensors that wait for events to reactivate the system, e.g. when a phone call is arriving, the device is automatically powered up again. In the *Hibernation* state no power is consumed because all functionality is freezed and stored on a nonvolatile memory. The system can be manually reactivated by restarting the system again at the point of suspension.

Processing power or power consumption are very popular as the Top500 Supercomputers Site **?** demonstrates. Considering only the top performer is already a moving target because the processing power is developing so rapidly. Making a partition with say five performance classes would be very disputable and should be readjusted from year to year. Nevertheless "number crunchers" are indispensable in science for weather forecasting, computing complex ecological models, gene sequencing, in technical developments for the simulations of car crashes, climate influence of large buildings, and in the military and intelligence area for cracking cryptographic codes.

More intrinsic properties of hardware are their technology and their architecture. New technologies appear only at a rate of a couple of years or a decade. There is no need to change the criteria but to add new classes. These most often do not resemble much the classical von Neumann architecture (see Chapter 9). For high performance computer systems we find processors with a reduced, and simpler instruction set than the standard central processor unit (CPU). For the PlayStation 3 a highly parallel computing cell design was used to boost processing power for e.g. the reality-like three dimensional graphics rendering and display.

Not only the computing power is a distinguishing criterion. Embedded processors that control some technical equipment or react to the environment within certain time limits need to have realtime capabilities. This can be achieved by sensors and interfaces that are capable of interrupting the processor at any time. If the MP 3 player would not deliver audio data in regular time intervals you would notice a distracting distortion of the sound.

The computer that controls the air bag system or the motor management that controls the fuel injection need reaction times in a range from 50 to 500 μ sec. Realtime systems for production control provide reaction time of some 10 μ sec, and missile control requires a response time as low as 10 nanoseconds. The operating system has to guaranty the response time but precondition is that the hardware provides this capability.

Lets get precise. What do we mean by architecture in terms of a computer? In the construction of a building we all understand the word **architecture** as the shape, composition, design, and in general the art of constructing a building. More specifically, and in the figurative sense we mean the arrangement (shapes and relationships) of objects. If we apply this to the computer it means how the computer is built from components or parts. We have presented in chapter 9 the historical prototype architecture of modern computers. It consists mainly of an Arithmetic Logic Unit (ALU), Control Unit (CU), memory unit, Input/Output (I/O) Controller and a bus system that connects all four components.

Today, the situation is much more complex because more then one ALU, more than one bus system, and a hierarchy of memory units are used to build a computer. State-of-the-art computer systems have many registers and two or more ALUs, called **dual-core** or **multi-core processor**. This will be one criteria for classification. The other will be the degree of parallelism in terms of data streams, meaning that a single instruction can be applied at the same time to more than one memory location. This is only possible if there are parallel data paths (data buses) to the memory.

Having two criteria, single or multiple ALUs and single or multiple data streams, which are independent of each other, this spans a 2×2 matrix (see Figure 10.1) with four possible types of computer systems.

Michael J. Flynn proposed this classification of computer architecture that distinguished computers according to their ability to execute multiple instructions or work on multiple data streams at a time. According to this scheme we classify as

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

Figure 10.1 Flynn's classification of computer architecture

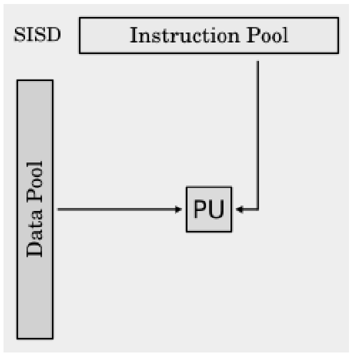

Figure 10.2 SISD schematic block diagram (PU = processing unit e.g. ALU)

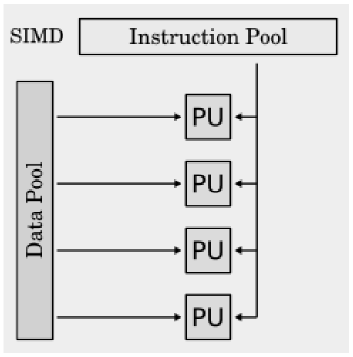

Figure 10.3 SIMD schematic block diagram (PU = processing unit)

- Single Instruction, Single Data stream (SISD)
  This are computers where both instructions and data are processed sequentially [10.2]. Examples are the early PC processor of Intel 808x.
- Single Instruction, Multiple Data streams (SIMD) This is a computer architecture which executes a single instruction to a set of data at the same time [Figure 10.3]. Vector processors as used in Gray's supercomputers[1] and graphics processors for video games are examples for SIMD.
- Multiple Instruction, Single Data stream (MISD)
  Multiple instructions operate on the same data [Figure 10.4]. This architecture can be used for fault tolerance or computing intensive tasks.

[1]manufactured by Cray Inc. (formerly Cray Research) **?**

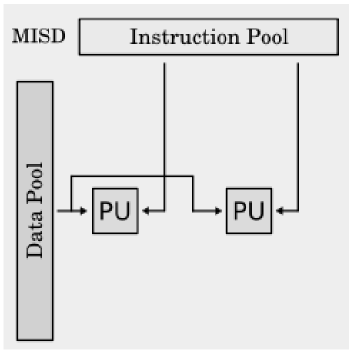

Figure 10.4 MISD schematic block diagram (PU = processing unit)

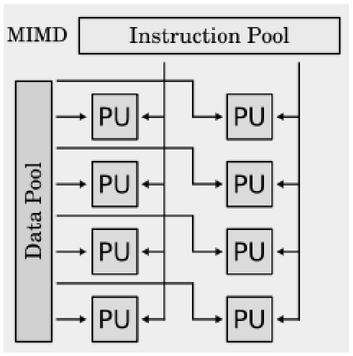

Figure 10.5 MIMD schematic block diagram (PU = processing unit)

Examples include the clock synchronous Stratus computer[2], space shuttle flight control and nuclear power plant control computers.

- Multiple Instruction, Multiple Data streams (MIMD)
  Multiple autonomous processors are simultaneously executing different instructions on different data [Figure 10.5]. We distinguish between closely coupled *Multiprocessor* systems and loosely coupled *Multicomputer* systems. The latter are multiple computers working together in distinction to a multiprocessor system having more than one processor but only one memory and one control unit.
  The inter-processor communication is established via high speed buses. In the case of multicomputer systems the inter-computer communication can be realized via *Local Area Networks (LAN)* if the computers are all in one building or at least on one property. If the computers are located in different cities the internet can be used for communication.

Even if this classification is rather rough it has been widely accepted over the last forty years. Not only parallelism is a criterion but also the size of the

[2]manufactured by Stratus Technologies, Maynard, MA **?**

instruction set and the complexity of the instructions are important as we will see in the next section.

10.2 CISC and RISC

Looking inside a processor, today's computers either use a complex or a reduced instruction set. At the beginning of digital computing the memory was so expensive that the manufacturers created processors with complex instruction sets that contained more logic for the same amount of memory. The drawback is that **Complex Instruction Set Computing (CISC)** requires for decoding and execution more time as a simple instruction. For instance a *negation* instruction has to

1. configure the ALU for negation
2. select a register as operand (r)
3. address a memory location (m)
4. load the memory content into the register ($r := m$)
5. wait for the load to complete
6. negate the register ($r := \neg r$)
7. store the register content back to memory ($m := r$)

This little example shows how many steps are involved in the negation of a number in memory. In the early times of computing it was nice to have powerful instructions as people were programming in machine or assembler language. The programmer could write the line

$$\text{NEG } m$$

instead of writing the above sequence of seven low level micro-instructions. A *micro-instruction* is a hardware implemented elementary operation in a processor that is executed in one time unit (*clock cycle*). A CISC processor instruction consists of a number of such micro-instructions. As the number depends on the instruction the time needed to execute an instruction varies from 1 to 20 time units.

In the mid 1980th the CISC design reached its limits because the processor speed mismatched the memory access time more and more. Please note that the ALU has to wait for the completion of the load operation. So not only that the processor had to execute a sequence of *micro-instructions* but it was necessary to wait for completion of the memory access.

In addition, it was noted that 80 % of all computations were using only 20 % of the machine instructions **?**, **?**. The processor designers turned their effort to **Reduced Instruction Set Computing (RISC)** as IBM called their CPUs that worked with a very limited set of simple instructions. The emphasis lies on simple instruction, not on the number of instructions. The most heavily used instructions were built into the hardware (*wired instructions*) and not composed of more elementary operations (*micro-instructions*) as in the above case of a *negation* instruction on a CISC processor. For simplicity and performance reasons the arithmetic and logic instruction of a RISC processor work on register data only. The benefit is that the operation does not have to wait for loading the data from memory or to wait until the data is stored back again. Take as example the *addition* of two integers. The ARM (*Acorn Risc Machine*) processor impements the ADD instruction using three registers r_0, r_1, r_2.

$$\text{ADD}\, r_0, r_1, r_2 \qquad // \; r_0 := r_1 + r_2$$

The loading of the registers and storing back the register content into memory is done with special LOAD and STORE instructions that access memory because arithmetic and logic instructions cannot access memory.

But how can we be faster, if two or more instructions are necessary for a RISC to do the same as *one* CISC instruction? Have we only shifted micro-code to the instruction level? The answer is to look ahead in the sequence of instructions and execute operations in parallel that do not conflict. An *addition* may be executed at the same time as loading data from memory into registers if the registers used by the addition are disjoint with the registers used for the data loading. Which makes it necessary to supply more registers for RISC processors.

As example lets take the Intel 8086 processor developed in 1978. The processor had 14 Registers and circa 120 instructions. The ARM (*Arcon* instruction set has 42 instructions, with several options each, e.g. conditional execution that access 16 general purpose registers and nearly 100 special registers for floating point operations, status and control registers, and WMMX registers for SIMD instructions that support multi-media applications.

Suppose we want to add two numbers a, b from memory and store the result in c and then compare the result with the number d. In order to execute this task in a CISC with two registers we use the instructions ADD a, b, STORE c and COMP c, d. These instructions translate to the following sequence of micro-instructions:

1. configure the ALU for the ADD instruction,
2. load number a into the first register,
3. load the second number b into the second register,
4. add both numbers and store the result in the first register $r_1 := r_1 + r_2$,
5. configure the ALU for the STORE instruction,
6. store the first register r_1 to memory location c
7. configure the ALU for COMP are,
8. load the value c to compare into the first register
9. load the value d to compare into the second register
10. compare both values $r_1 = r_2$.

These are ten steps. For simplicity each step shall need only one *clock cycle*. In the case of RISC we need only four cycles using four different hard-wired and overlapping instructions as we see in the following instruction sequence:

1. LOAD number a into register r_1,
2. LOAD the second number b into r_2,
3. ADD both numbers $r_3 := r_1 + r_2$ and LOAD the value d into r_4,
4. COMP are both values $r_3 = r_4$ and STORE the previous result r_3 into c .

After loading the two numbers a and b and while executing the addition we can already load the data d to compare in register r_4. In the forth step the result of the addition can be compared with the just loaded value in r_4 while the result of the previous addition in r_3 is stored into memory location c.

This means that we save six cycles as result of the parallel execution of simpler instructions and the availability of more registers in the case of RISC. In practice the instructions often need more than two cycles so that an even higher degree of parallelism will be possible as we can see in Figure 10.6 from the next section.

The RISC architecture influences the programming. While the small number of instructions and its simplicity facilitates programming the increased number of registers and the low level instructions complicate programming at the same time. The programmer has to decide which registers to use and which variables to keep in the register for further use and which should be written back to memory.

In the last twenty years chip integration density has increased so that CISC architectures can include more registers and use pipelining, too. On the other hand the higher integration allowed RISC architectures to increase the number of hardwired instructions and to add dedicated processors for floating point arithmetic and multimedia encoding and decoding.

This development tends to converge both technologies and makes it more difficult and less meaningful to distinguish processors in the above way. Todays (2009) processors use two to eight ALUs (*cores*) and still support the old complex instruction set as well as RISC type instructions.

10.3 Pipelining and Parallelization

Pipelining is like an assembly line. At each workplace a worker performs certain activiities to assemble part of the product. The time to accomplish is limited and after the time limit is reached the partially fabricated product is handed to the next worker. Each of the workers work in parallel and when the time limit is reached, each partially assembled product is handed over to the next workplace. After a number of work steps the product reaches its final workplace and will be finished.

If there are n workplaces then we have n products in the assembly process. With each time limit the product in the last assembly step will be finished, and all others will have progressed by one step. With this concept of building a product the number of products produced in a given time period does not

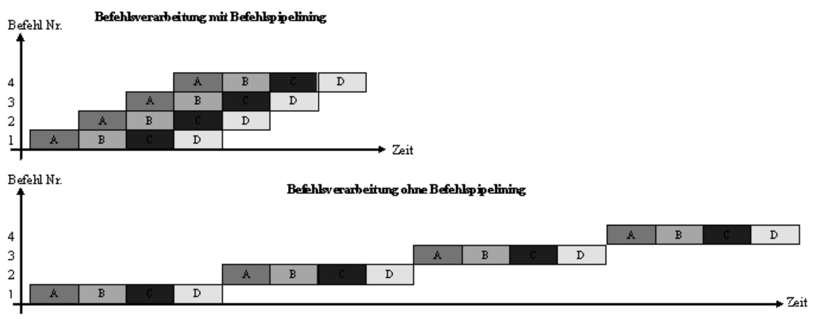
Befehl Nr.
Befehlsverarbeitung mit Befehlspipelining
Zeit
Befehl Nr.
Befehlsverarbeitung ohne Befehlspipelining
Zeit
A
B
C
D

Figure 10.6 A four step instruction pipeline

depend on the number on workplaces but only on the longest time limit *t* (clock cycle) to finish an assembly step, because at each clock cycle one product is finished. Therefore the maximum time needed at any workplace should as short as possible and should not vary much from on workstep to the other.

As consequence finishing a product every *t* seconds requires to divide the work into work steps where each workstep requires not more than t seconds.

In a processor this concept is applied to the execution of machine instructions. The division of a machine instruction into a series of processing steps where each step may be executed at the same time with other instructions is called **pipeline** processing. The time used for an instruction step is driven by a clock generator. At each clock tick a machine instruction can be accomplished while some others are still in progress. Figure 10.6 illustrates the mechanism with a four step RISC processor. The four processing steps are instruction Fetch (A), Decode (B), Execute (C), Write (D).

A problem arises when an instruction needs the result of a previous operation. In this case the processor has to wait for the result. Reordering of instructions may also solve the conflict. Waiting is achieved by *no operation (NOP)* instructions that are inserted to fill the time gap.

A branch instruction invalidates all instructions in the pipeline. It results in a so called *pipeline stall* because processing has to wait until the next instructions are loaded to execute. Therefore strong effort is made to predict the branch decision and load the instructions in advance according to the prediction. If the prediction comes true, the processing can continue without delay.

Another problem results from situations where the hardware cannot support the overlapping of instructions due to hardware limitations, e.g. data path restriction or control unit restrictions.

These three types of problems are collectively called *hazards* which lead to a stall. But there are also situations not related to RISC which lead to stall a processor. A processor may need to wait for input or output to take place. If instruction caching is used (see Chapter **??**) and the next instruction is not in cache the processor has to wait until the instructions is read from memory or "paged in" from disk.

We have seen that overlapping instructions in the RISC processor yield higher performance. Therefore lets have a closer look at another form of parallelism.

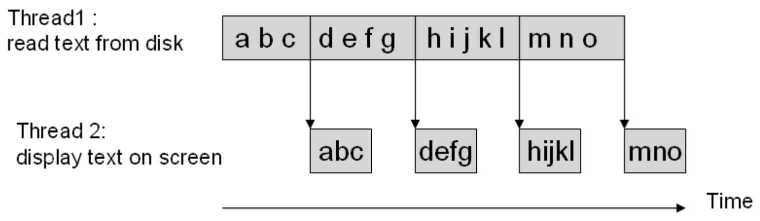

Figure 10.7 parallel processing example

A similar, yet different approach to boost performance is working at different instructions or at different data at the same time using multiple processors. We call this **parallelization** of instructions resp. of data. This is a higher level of parallelism than pipelining. Pipelining is a kind of overlapping instructions in a clock synchronous manner in one processor whereas parallelization of instructions means executing unrelated instructions at the same time in different processors.

The parallelization of processors has its pendant in the production process. The production rate of an assembly line is limited by the time limit (clock cycle) for the largest work step. To overcome this limitation we could use more than on assembly lime, thus multiplying the product output rate. This is exactly the approach when we use multiple processors in parallel.

If we have more than one ALU we can process two or more execution threads at the same time. A **thread** is a sequence of operations that are related (like a sequence of work steps in an assembly line). A process consists of at least one execution thread. All threads of a process share the same resources.

Each thread may run in any ALU which means that threads can run in parallel provided they do not want to use the same resources at the same time. If two threads share the same resources (e.g. belonging to the same process) they need to synchronize (see Chapter **??**) from time to time. This is necessary when one thread needs the results from an other thread. As example take a document reader process that consists of two threads, one reads a document character by character from disk and the other displays it on the screen. The disk reading can continue while the characters are being displayed on the screen. This is comfortable for the user as he can see already the first page of a document when the reading of the document is still in progress. On the other side the display thread has to wait for the disk before it can display anything. If the display is faster than reading the data from disk we have the situation as sketched in Figure 10.7. Reading and displaying the data requires in this case the same time as reading alone plus one extra display time because displaying is done during reading time except for the last data portion.

Displaying of data is also a good example for parallelization of a data stream. You can think of the screen data as a matrix of memory cells. The same operation has to be executed on many cells. If your ALU is able to access and process more than one memory cell at a time with the same instruction we have a kind of SIMD processor as in Figure 10.3. We can read resp. display an array of data simultaneously.

The most common use of parallel processing is the case in multitasking. Instead of switching the processor between different tasks a processor can be

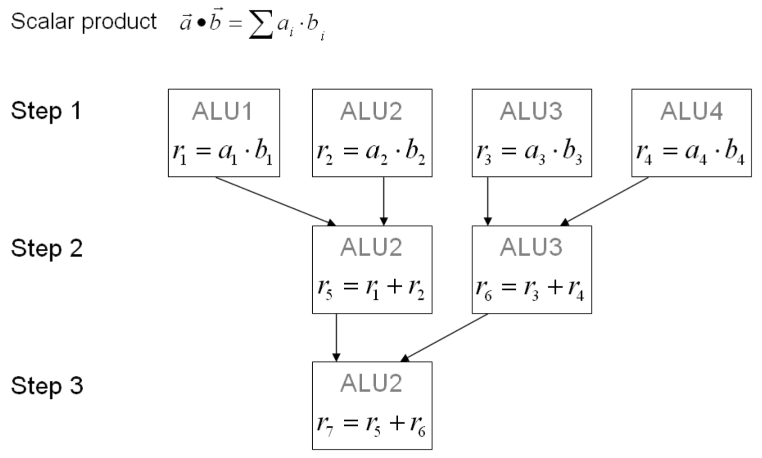

Figure 10.8 Parallel algorithm for a four dimensional scalar product

dedicated to a task. This avoids task switching overhead and results in better performance. The *dispatcher* (see Chapter **??**) assigns the physical resources like the ALU to a process. The process may execute until completion if no other process is competing with the same ALU.

Parallel processing on a process level has no impact on programming. But if we want to work in parallel on one task we need parallel algorithms instead of sequential algorithms like those discussed in Chapter **??**. Lets assume that we want to calculate the scalar product $\vec{a} \bullet \vec{b} := \sum a_i b_i$ of two four-dimensional vectors $\vec{a}$ and $\vec{b}$. This requires four multiplications and four additions. A sequential algorithm would need 8 time units assuming for simplicity that the arithmetic operations need one time unit each. If we have an quad-core processor all multiplications could be executed simultaneously in one time unit. In the next step we could add the products two by two and in the third step we yield already the result as illustrated in Figure 10.8.

It is important in technical and scientific calculation to be able to find parallel algorithms. However this is not a very promising task. Often we have work sections that cannot be parallelized. If half of the work steps are serially executed the total execution time is at least half of the time that would be needed for a purely sequential execution no matter how many processors we have. This is essentially the statement of *Amdahl's Law*. If $1/n$ of the work steps need sequential execution and these need $(1/n)$th of the sequential execution time than there is only $(n-1)/n$ time units left for parallelization. This means we do not even reach a speed-up factor of n. Another limiting factor of parallelization is the communication and synchronization overhead. Synchronisation leads to waiting for a resource to become available as pointed out in the case of pipelining. But also in the case of parallel-threading or parallel-processing there is a need for exchanging information between processors. This can be done by sharing memory or registers which leads again to synchronization problems. If the communication is realized with message passing the cost is much higher than sharing memory and the data bus may become a bottleneck.

The highest level of parallelization is on the computer level. Here we have a network of computers that work on a common task. This might look more efficient than parallel-processing as there is no sharing of resources. However the communication pathes for n computers directly interconnected with each other need $n(n-1)/2 \approx n^2/2$ data lines. This is not feasible for larger n which motivated network topologies that need less lines, like bus, ring or star structures (see Chapter **??**). But the simple network topologies come with the prize of a bottleneck on the communication line and on the pass through computers. The communication between any two computers has to share the same line in case of a bus structure or need to pass the data through other computers which generates a significant overhead.

In this section we have seen three levels of parallelism, pipelining, parallel threading or parallel processing, and parallel computing with decreasing coupling strenght. While in the case of pipelining a lot of resources including registers are shared, parallel threading shares memory only, and parallel computing share nothing in a fully connected network.

Todays processor manufacturers tend to integrate more ALUs (cores) into their processors. Latest products include eight cores. For a given number of cores the models mainly differ in term of cache size and the degree of dedicated data and control pathes (buses).

10.4 SSD Technology

Beside processor and memory technologies external mass storage is a main component of a computer system. Nonvolatile mass storage is used for storing large amount of data, notably documents, pictures, audio and video files. In addition to data all installed programs are stored on nonvolatile mass storage.

For nonvolatile mass storage the hard disc drives have established itself as first choice. The data bits are stored on a rotating magnetic disc surface (see Figure 9.2 in Chapter 9). Lately hard disc drives are being replaced step-by-step by *solid state drives (SSD)*. The name is kind of misleading as SSD are storage devices with no movable components. This make them very robust against mechanical shocks.

There are two technologies available for solid state drives.

The first is similar to random access memory (RAM) as explained in Section 9.6.1. It uses electronic parts that store bits in microscopic semiconductor capacities. SSD that use the *SDRAM (Synchronous Dynamic Random Access Memory)* technology provide fast access similar to main memory. One storage cell of the dynamic RAM consists of a *field effect transistors (FET)* and a capacity. FET and capacity are manufactured in different technologies using etching and doping of a silicon substrate (Figure 10.9). The capacity acts as storage cell. When it is loaded it represents bit “1” , and when discharged it represents bit “0”. The FET is used as an electronic switch to read or write the storage cell. The cell is selected by applying positive voltage on the word address line V_{WL} and the Gate G (Figure 10.10) which make the FET conducting (low resistance between drain D and source S). Depending on the charge of capacity C the voltage difference between V_{BL} and V_{PI} is low or high representing bit 0 or 1.

Because of leak current the capacity needs to be recharged from time to time. Usually this is done in regular time units, the *refresh cycle*. This characteristic fact

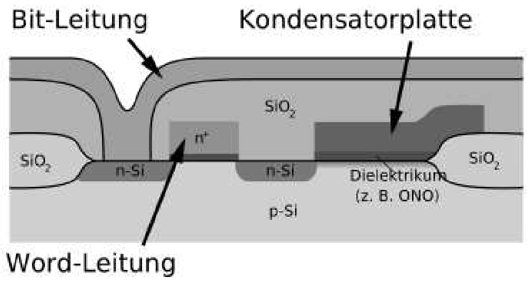

Figure 10.9 Vertical cut through the wafer of a DRAM cell

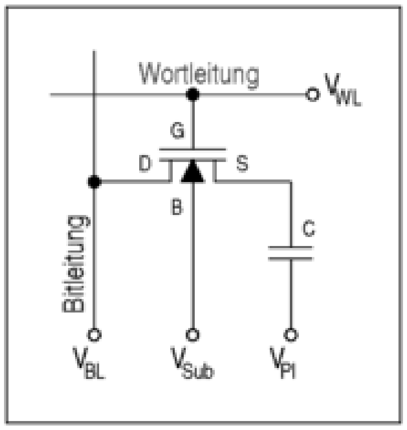

Figure 10.10 Accessing a DRAM cell

distinguishes Dynamic RAM (DRAM) from Static RAM (SRAM) that needs no cell refresh. The cyclic recharging of memory capacities is called *memory refresh*. This makes permanent power supply necessary if the data should persist.

Instead of using SDRAM technology a cheaper technology called *EEPROM (Electrically Erasable Programmable Read-Only Memory)* has become popular in recent years. The name suggests that this technology provides a non volatile storage that can be electrically erased if desired. The EEPROM contains FETs in floating gate technology. The gate is perfectly isolated from the transistor's source and drain. To store one bit on the gate either a quantum mechanics tunnel effect (the *Fowler-Nordheim-effect*) or the *hot electron injection* is used to bring some electrons (electrical charge) onto the gate. Hot electron injection is possible by acceleration of electrons through a voltage of typically 10 - 18 V to gain sufficient energy to surmount the isolation potential barrier. Using the Fowler-Nordheim tunnel effect requires less voltage as the isolation potential barrier is "tunneled" by the electrons to reach the gate.

When the gate is loaded, the FET gets unconductive while it is conductive in the discharged state. This mechanism allows to store one bit (conductive - unconductive) on each FET. Loading and discharging is much slower than for SDRAM technology but the benefit is the nonvolatile character of the storage device. Discharging is casually called "flashing" the memory giving this technology its common name *Flash Memory*. Flash memory in the form of a

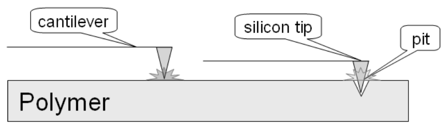

Figure 10.11 Accessing a DRAM cell

"memory stick" is very popular as mobile SSD as it comes with a *USB (Universal Serial Bus)* interface for easy use. The same technology is used for MP3 payers as the data capacity is high and the memory module is very robust and does not need power when not in use because no memory refresh is needed.

Hard disk drives provide low cost storage. In the year 2008 the storage cost for one gigabyte (1 GB) declined to about 0.3 $ for hard disks compared to 10 $ for flash memory and 60 $ for SDRAM technology. SDRAM has about 100 times shorter access time than hard disc drives.

10.5 Millipede Storage Technology

In the early stages of computing punch cards where used to safely store data or programs outside the computer. These cards contained the data in form of holes. In principle the same concept but on a microscopic scale is used in the *Micro-Electro Mechanical Systems (MEMS)* technology. This technology of the very small is not limited to storage devices but applies to any nano-scale electromechanical machines. Those machines are expected to be used to introspect humans through inserting a nano-device into the blood circulation in order to inspect and repair human organs or dispose medicin locally. MEMS are made up of components between 1 to 100 micrometers in size (i.e. 0.001 to 0.1 mm). The total size of a MEMS is typically less than 1 millimeter. Our focus is on MEMS that are able to store digital data.

In the year 2000 researchers[3] from the Carnegie Mellon University proposed the replacement of the rotating disc paradigm in favor of MEMS-based storage system. It took until 2005 when IBM announced a nano-mechanical device with a storage density of more than one terabit (1000 GB) on a square inch. This is about 4 times the density of a magnetic hard disc. The base material used is a polymer film where pits of approximately 10 nanometer in diameter were melted into the material. The pits are sensed with a silicon tip residing on a little cantilever (see Figure 10.11). The apex of each tip has a radius on the scale of a few nanometers to sense nanoscopic pits. "Reading" the pit is done by measuring the conductivity change rate after heating the pit up to 300° C. The surface is larger when a pit is present than without pit. In case of a pit the temperature decreases faster because of the larger area and hence the conductivity gradient is higher than without pit.

A pit is "burned" into the polymer by heating up the tip at 400° C. The polymer is locally softened by the heat just above the glass transition temperature and simultaneously applying a little force to create a nano-scale indentation. The indentations remains after cooling down of the polymer.

[3]R. Carley, G. R. Ganger, and D. Nagle

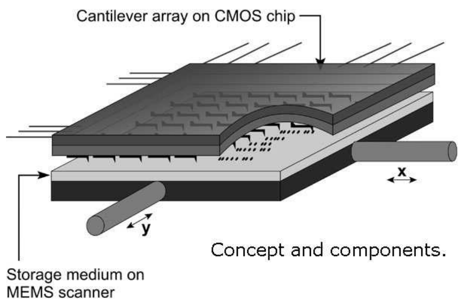

Figure 10.12 Cantilever array

However, the indentation creates a tension that can be used to reverse the indentation at a later time.

The pit can be filled again by reheating without indentation force. Due to the elastic tension the pit retrieves its original shape as before the indentation. There is still ongoing research on the effects of repeated erasing and re-writing.

To ensure high data transfer rates not only one bit or byte is read or written at one instant of time. IBM uses a cantilever matrix of $64 \times 64 = 4096$ tips leading to a data rate of 400 megabit per second. The thousands of cantilevers gave reason for IBM to call the project “millipede” (Figure 10.12).

The cantilevers used in the matrix have three terminals, with separate heaters for reading and writing, and a capacitive platform for electrostatic actuation of the cantilevers in the z-direction. Movement of the polymer storage medium relative to the cantilever array is achieved using a silicon-based x/y microscanner. Positioning information is provided by two pairs of thermal position sensors. The sensors consist of thermally isolated, resistive silicon strips. The strip is heated by current flowing through the resistor. A fraction of this heat is conducted through the ambient air into the scan table, which acts as a heat sink. Displacement of the scan table gives rise to a change in the efficiency of this cooling mechanism, resulting in a change in temperature of the heater. These sensors provide a resolution of less than 2 nm.

The commercialisation of MEMS storage devices has not yet begun (2008) but experiments demonstrated in benchmarks approximately five times better performance over conventional hard disc drives[4].

10.6 CD-ROM and DVD

The *Compact Disc (CD) Digital Audio* is an optical media for storing music in digital format. It was developed by Philips and Sony and released as a standard to the market in 1982 **?**. The audio signals are sampled and digitally coded as explained in Section **??**. The disc is divided into a maximum of 99 tracks. A track corresponds to a piece of music, i.e. a song. Each track is divided into blocks of 1/75 second. The addressing is a time code in the format *mm.ss.bb*, where the block number *bb* ranges from 0 to 74 and *mm* and *ss* represent minutes and seconds, respectively. This allows theoretically to address 100 minutes of music.

[4]Feng Wang et al., Using MEMS-based Storage to Boosting Disk Performance **?**

However a typical CD Digital Audio has a capacity of 70 minutes. Each block contains 2353 bytes which correspond to a bit rate of 1411,2 kbit/s. This data rate is required for a sampling rate of 44,1 kHz with a 16 bit resolution which provide an audio bandwidth of 5 Hz to 20 kHz with a dynamic range of 96 dB. The addressing scheme is therefore mainly suitable for musical purpose. Three years later the CD standard was extended in the *Yellow Book* with addressable blocks and error correcting code. This makes the CD usable for computer data storage. If we use the same encoding as for an audio CD the capacity of a data CD would be about 650 MB.

10.6.1 Compact Disc Read-Only Memory

After the Yellow Book specification the ISO 9660 **?** standard defined a file system with filenames containing not more than 8 characters plus a file name extension of 3 characters (DOS format). In addition, the standard requires that a file need to be stored contiguously. Compact discs following this standard are known under the name *Compact Disc Read-Only Memory (CD-ROM)*.

The disc platter itself is made of thermoplastic polycarbonate, a synthetic material that is sensitive to heat and cold. The material should not be exposed to temperatures below -40° C because the material perishes and at temperatures higher than 70° C it will deform and melt.

The data layout of a data CD as defined by ECMA-130 **?** consists of one or multiple sessions. Each session contains a *lead-in*, a user data track, and a *lead-out*. The lead-in contains the table of contents (TOC) for this session and the starting address for the next session. If the CD is finalized the next session address is empty. The lead-out contains time information of the session, e.g. duration, timing information, etc.

The data is stored linearly in form of a spiral. Along the spiral track are tiny indentations (*pits*) and *land* in between pits. The data is stored from inside to outside edge with a constant bit density. From this follows that the rotation speed of the disc is faster if the data is located near the center. The CD-ROM is produced industrially by injection moulding the polycarbonate at high temperature using a *stamper* disc that contains the inverse coding. Pits and lands do not directly represent bits. The encoding used is *non-return-to-zero inverted (NRZI)*. With this encoding bit “1” is represented by a change from land to pit or the reverse, while no change indicates bit “0”.

This change is measured by the changing reflection of a red laser beam. The distance between the reflective coating of lands to pits is $\lambda/4$ which results in a phase shift of $\lambda/2$ for the reflected beam. The beam and the reflected beam interfere in such a way that the resulting intensity is practically vanishing at the pits-lands transition. Figure 10.13 illustrates the situation with an NRZI-encoding example.

To individually produce a CD-ROM you need a CD writer and a blank CD, called *CD Recordable (CD-R)*. The base material is the same as for the CD-ROM. The disc contains a spiral groove in order to help the CD writer to direct the laser beam for writing. So a blank CD-R is not really “empty” but it has not yet any pits or lands. Inside the disc there is a thin layer of organic dye and a reflective silver coating. The coating is protected by lacquer and sometimes with an additional protective white paint to make the surface ready to write on it or for decorative printings.

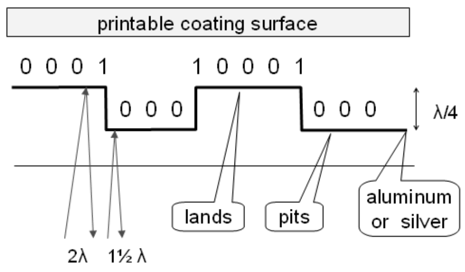

Figure 10.13 NRZI coding example on a CD-ROM

The data is written ("burned") into the disc plattern with a strong laser pulse. The laser pulse energy heats up the dye in a small area (pit). The dye changes its absorption characteristics at the pit location. When the disc is read with low power laser beam the reflection changes on each pit-land edge in the same way as for the NRZI encoding of a CD-ROM. This allows ordinary CD player or CD-ROM drive to read a CD-R even if there are no cavities (pits), but the reflection of the laser beam is similar. The reading does not affect the dye because the laser power for reading is much lower than for writing.

Sometimes it is not possible to copy a CD because the CD has a copy protection. In the simplest case the CD contains code errors or do not conform to the specification of the Red Book. In such a case the CD drive cannot read part of the data and the CD burner refuses to record corrupted data. Playing a CD on the other hand is tolerant against data read errors as such a CD can still be played or read with the exception of the corrupted block.

10.6.2 Rewritable Compact Disc

A ReWritable CD (*CD-RW*) has a similar structure as the CD-R but instead of the dye there is a reflective layer of silver-indium-tellurium alloy (Figure 10.14). This poly-crystalline alloy reflects the laser beam. When a strong laser pulse heats up the alloy to over 400° C the alloy changes structure to an amorphous state (pit). In this state the alloy is transparent and the beam traverses the alloy and is absorbed in the coating of the disc. The change in reflection is not produced by inference but by different reflective material.

To erase a recording the laser beam heats up the pit area to a temperature of about 200° C. During this annealing temperature the alloy regains its poly-crystalline structure. To rewrite a CD-RW it must be erased first. It may be rewritten about 1000 times. The life expectancy is with 20 years significantly less compared to a CD-ROM with an expected lifetime of 50 years. The lifetime depends highly on the environmental influences, temperature, chemicals, dirt or mechanical damage, etc.

As the CD-RW has much lower reflection signals it does not meet the specifications of the Red and Orange Book. This can lead to problems with older CD players and CD-ROM drives if they do not show the *MultiRead* compatibility specification.

It should be noted that CD-R can add new data by adding a session if the disc has not been already finalized. Adding a new session does not erase the already present data. The old data is still there plus the newly recorded data.

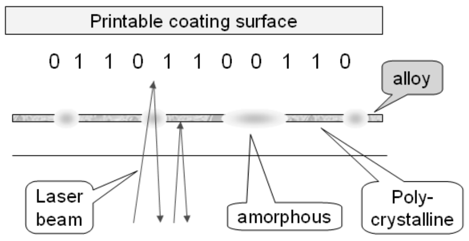

Figure 10.14 Vertical cut through a rewritable CD

Some CD-burner seem to allow to "delete" data from one session. In reality the data is made only inaccessible, because the table of content (TOC) is written in the newly recorded session which replaces the old TOC.

In contrast to CD-R the CD-RW really erases the whole content of a CD and allows to rewrite the storage medium.

10.6.3 Digital Versatile Disc

Some years after the introduction of the CD Audio the same disc platter was used to store video information. But the capacity was not sufficient to store a full Length movie in high quality. This lead to the development of the *Digital Versatile Disc (DVD)* with a capacity of 4.38 GB (= 4380 MB) for a 12 cm standard disc. The DVD comes in different physical sizes but the most common one has the same size as the CD. In addition to the different physical sizes there are half a dozen formats for the encoding to support special features for video, random access, time search, recordable, rewritable, double layer, etc.

The production process and the encoding of a DVD is similar to the CD. Basically every thing is more dense. The spiral track has less distance to its neighbors, the pits and lands are smaller. This is possible by using a laser with a wavelength of 650 nm compared to 780 nm for a CD. The shorter wavelength is better focusable and produces smaller pits. The pits and lands are closer to the downside of the disc to reduce the diffusion of the laser beam. All this results in a 6 times higher capacity than the CD. As side effect of the higher data density the data transfer rate increases proportional with the capacity.

The high encoding density makes it more difficult to produce and detect very short bit change sequences like 1010101. To avoid reading errors the bit stream is transformed in a way that between two "1" bits there are at least two "0" bits and at most 10 "0" bits. This is known as *run length limitation*.

When using the DVD as video storage the data is usually MPEG-2[5] encoded and using a resolution of usually 704 × 576 (PAL) or 704 × 480 (NTSC). This is sufficient to store a full size movie (120 min) on a DVD. The capacity of a DVD is still not enough to store a full size movie in a high resolution like the HDTV with an resolution of 1920 × 1080 pixels. This made it necessary to introduce a new CD standard with a capacity of a least 20 GB.

[5]see Chapter **??**

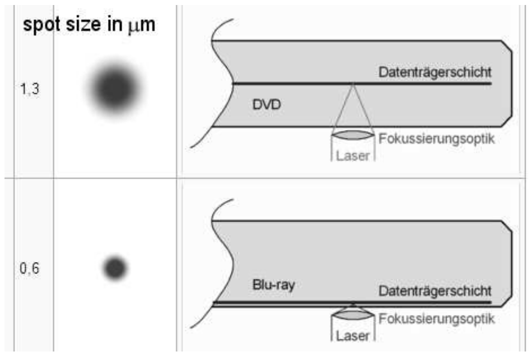

Figure 10.15 Spot size comparison between DVD and Blu-ray Disc

10.6.4 Blu-ray Disc

From 2005 to 2008 three competing standards were developed as successor of the DVD. The HD DVD (High Density Digital Versatile Disc) was developed with capacities between 15 and 30 GB. The Versatile Multilayer Disc (VMD) was essentially a DVD with 10 storage layers resulting in a capacity of approximately 40 GB. The developer of this storage medium, New Medium Enterprises, claims that their technology has the potential for a 100 GB capacity. And the third technology, called *Blu-ray Disc*, is supported by Panasonic, Philips, Sony, and others. After the retreat of the HD DVD from the market it seems that the Blu-ray will become the next standard.

The name Blu-ray stems from the blue laser beam with a wavelength of 405 nm. The spot size of a focused light depends on the wavelength and the aperture of the lens. The data layer is very close to the surface of the disc and the laser optic is designed for a high aperture. This allows a better focussing of the laser beam. By decreasing the wavelength and increasing the aperture the spot size came down from 1.3 μm (DVD) to 0.6 μm for Blu-ray (see Figure 10.15). This means that the blue light is about two times better focusable than the red light.

With this tiny spot size the data density increased to 25 GB for a single sided disc. With single drive speed the Blu-ray has a data rate of 4.5 MB which is 4 times faster than the DVD. Because the data layer is closer to the surface a special scratch resistant coating is necessary.

10.7 Summary

- At the beginning of this chapter we presented Flynn's classification scheme for multiprocessing architectures in terms of parallel execution of instructions and parallel data access. We distinguished between multi-processor and multi-computer architectures. This kind of classification was justified by comparison with other criteria.
- A closer look was taken into RISC technology, especially to pipelining. We showed in detail how RISC technology outperforms the traditional CISC. From this intra-processor parallelism we moved to higher level

of parallelism as multi-processors (multi-core) and multi-computer architectures. The appropriate programming models are multi-threading and multi-processing. This lead to the conclusion that parallel algorithms are necessary to exploit parallelism on this level. However, Amdahl's law says that if $1/n$ of a process is not parallelizable no matter how many processors we use the maximum performance gain will be limited by a factor of n.

- The highest level of parallelism considered was network computing providing the weakest form of coupling. It does not share any computing resources except from the communication lines for certain topologies. The communication overhead is a limiting factor because it increases in square order with the number of computers.
- In the following sections state-of-the-art storage technologies were presented. Solid state drives (SSD) are considered as a shock resistant, high performance replacement of hard disc drives. Two technologies are promising: using SDRAM for high end performance and Flash technology for low cost nonvolatile storage. During a transition time hybrid storage drives may coexist.
- As alternative external mass storage the MEMS technology developed in the "millipede" project of IBM at Rüschlikon, Switzerland, set itself apart with an unmatched storage density of one terabit per square inch. This technology uses nanoscale indentations in a polymer foil to store binary data. The first commercial product is still awaited. The data is sensed in parallel with an array of 4096 tiny cantilevers which was the inspiration for the project's name.
- Beginning with the CD Audio and the CD-ROM for digital data we followed the development of optical storage media to its latest product the Blu-ray Disc. The capacity increased during this time from 650 MB to more than 25 GB. The different opto-mechanical characteristics and materials are presented and its functions explained in its context.

Review Terms

- Computer classification
  - Single Instruction, Single Data stream (SISD)
  - Single Instruction, Multiple Data stream (SIMD)
  - Multiple Instruction, Single Data stream (MISD)
  - Multiple Instruction, Multiple Data stream (MIMD)
- Computer architecture
  - Complex Instruction Set Computer (CISC)
  - Reduced Instruction Set Computer (RISC)
- Pipeline
  - micro instruction
  - hazard
  - clock cycle
  - NOP
- Parallelization

  - ○ thread
  - ○ synchronization
  - ○ multithreading
  - ○ multiprocessing
  - ○ dispatcher
  - ○ Amdahl's law
- Solid State Drive (SSD)
  - ○ Synchronous Dynamic Random Access Memory (SDRAM)
  - ○ Electrically Erasable Programmable Read-Only Memory (EEPROM)
  - ○ hot electron injection
  - ○ Flash Memory
  - ○ Universal Serial Bus (USB)
- Micro-Electro Mechanical Systems (MEMS)
  - ○ cantilever matrix
  - ○ silicon tip
  - ○ polymer storage medium
  - ○ Central Processing Unit (CPU)
  - ○ Bus subsystem
- Optical Storage Media
  - ○ Compact Disc (CD)
  - ○ CD Read-Only Memory (CD-ROM)
  - ○ No-Return-to-Zero Inverted (NRZI)
  - ○ CD Recordable (CD-R)
  - ○ CD ReWritable (CD-RW)
  - ○ Digital Versatile Disc (DVD)
  - ○ Blu-ray Disc

Exercises

10.1 List at least ten devices that probably contain a processor. For each device argue which indications do you have your assumption.

10.2 Which criteria are important for embedded computing devices?

10.3 Define the term "architecture" in the context of computing.

10.4 To which type of processor belongs to a multi-core processor in Flynn's taxonomy?

10.5 Argue, which of the four processor types has the highest performance potentials?

10.6 What distinguishes a multiprocessor from a multicomputer?

10.7 Show with an example the performance gain using pipelining.

10.8 What kind of "hazards" do you know with pipelining?

10.9 Which are the key factors for the pipelined execution performance?

10.10 What is the algorithmic prerequisite for parallel processing?

10.11 Explain the consequences of Amdahl's law.

10.12 What stands the acronym SDRAM for?

10.13 Are SDRAM volatile or non-volatile storage components?

10.14 Describe the erasing mechanism for EEPROM components.

10.15 What kind of storage medium is used for the “millipede” MEMS?

10.16 Describe the reading mechanism of a “millipede” device.

10.17 How much is the capacity of a DC-ROM, a DVD, and a Blu-ray Disc?

10.18 Why can you only write once on a CD-R but many times on a CD-RW?

10.19 Why does a CD turn at a decreasing speed when reading a file or playing a piece of music?

Bibliographical Notes

Hans W. Meurer **?** has documented the development of supercomputers over the last 15 years. The list of the top 500 supercomputers is published and updated every half year at http://www.top500.org **?**.

John Cocke, the Turing Award winner of 1987 **?**, found that most of the processing used only 20 % of the available instruction of a computer. His vision in 1965 was to build IBM’s first Advanced Computing Systems (ACS) which lead to the development of RISC System/6000 (later named RS/6000) **?**. The Pareto rule (80 % completness requires only 20 % of effort) was confirmed for the utilization of processor instructions with a case study by C. Foster, R. Gonter, E. Riseman **?**.

Millipede technology has been developed by IBM Research at Rüschlikon, Switzerland **?** in 2005. Its mechanism and physical architecture is explained on the Millipede project web site **?**.