Chapter 2
Heterogeneous Multicore Architecture

2.1 Architecture Model

In order to satisfy the high-performance and low-power requirements for advanced embedded systems with greater flexibility, it is necessary to develop parallel processing on chips by taking advantage of the advances being made in semiconductor integration. Figure 2.1 illustrates the basic architecture of our heterogeneous multicore [1, 2]. Several low-power CPU cores and special purpose processor (SPP) cores, such as a digital signal processor, a media processor, and a dynamically reconfigurable processor, are embedded on a chip. In the figure, the number of CPU cores is *m*. There are two types of SPP cores, SPP_a and SPP_b, on the chip. The values *n* and *k* represent the respective number of SPP_a and SPP_b cores. Each processor core includes a processing unit (PU), a local memory (LM), and a data transfer unit (DTU) as the main elements. The PU executes various kinds of operations. For example, in a CPU core, the PU includes arithmetic units, register files, a program counter, control logic, etc., and executes machine instructions. With some SPP cores like the dynamic reconfigurable processor, the PU executes a large quantity of data in parallel using its array of arithmetic units. The LM is a small-size and low-latency memory and is mainly accessed by the PU in the same core during the PU's execution. Some cores may have caches as well as an LM or may only have caches without an LM. The LM is necessary to meet the real-time requirements of embedded systems. The access time to a cache is non-deterministic because of cache misses. On the other hand, the access to an LM is deterministic. By putting a program and data in the LM, we can accurately estimate the execution cycles of a program that has hard real-time requirements. A data transfer unit (DTU) is also embedded in the core to achieve parallel execution of internal operation in the core and data transfer operations between cores and memories. Each PU in a core processes the data on its LM or its cache, and the DTU simultaneously executes memory-to-memory data transfer between cores. The DTU is like a direct memory controller (DMAC) and executes a command that transfers data between several kinds of memories, then checks and waits for the end of the data transfer, etc. Some DTUs are capable of

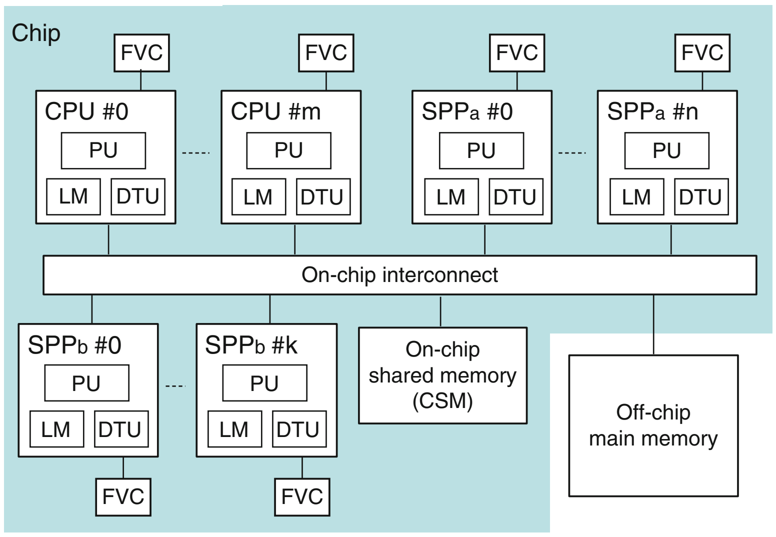
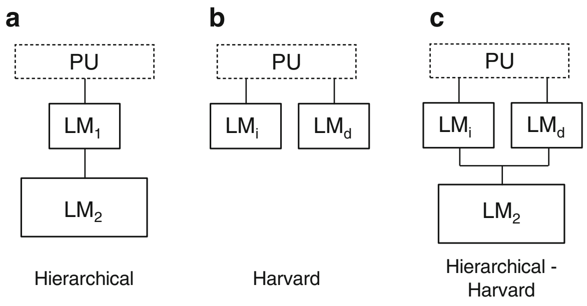


Fig. 2.1 Heterogeneous multicore architecture

command chaining, where multiple commands are executed in order. The frequency and voltage controller (FVC) connected to each core controls the frequency, voltage, and power supply of each core independently and reduces the total power consumption of the chip. If the frequencies or power supplies of the core's PU, DTU, and LM can be independently controlled, the FVC can vary their frequencies and power supplies individually. For example, the FVC can stop the frequency of the PU and run the frequencies of the DTU and LM when the core is executing only data transfers. The on-chip shared memory (CSM) is a medium-sized on-chip memory that is commonly used by cores. Each core is connected to the on-chip interconnect, which may be several types of buses or crossbar switches. The chip is also connected to the off-chip main memory, which has a large capacity but high latency.

Figure 2.1 illustrates a typical model of a heterogeneous multicore architecture. A number of variations based on this architecture model are possible. Several variations of an LM structure are shown in Fig. 2.2. Case (a) is a hierarchical structure where the LM has two levels. LM_1 is a first-level, small-size, low-latency local memory. LM_2 is a second-level, medium-sized, not-so-low-latency local memory. For example, the latency from the PU to LM_1 is one processor cycle, and the latency to LM_2 is a few processor cycles. Case (b) is a Harvard type. The LM is divided into an LM_i that stores instructions and an LM_d that stores data. The PU has an independent access path to each LM. This structure allows parallel accesses to instructions and data and enhances processing performance. Case (c) is a combination of (a) and (b). The LM_i and LM_d are first-level local memories for instructions and data, respectively. LM_2 is a second-level local memory that stores both instructions and data. In each case, each LM is mapped on a different address area; that is, the PU accesses each LM with different addresses.

Fig. 2.2 Structures of various local memories



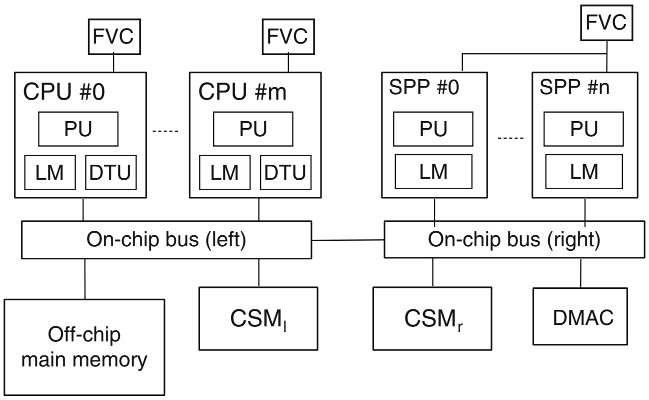


Fig. 2.3 Example of other heterogeneous multicore configurations

In Fig. 2.3, we can see other configurations of a DTU, CSM, FVC, and an on-chip interconnect. First, processor cores are divided into two clusters. The CPU cores, the CSM_l, and the off-chip main memory are tightly connected in the left cluster. The SPP cores, CSM_r, and the DMAC are nearly connected in the right cluster. Not every SPP core has a DTU inside. Instead, the DMAC that has multiple channels is commonly used for data transfer between an LM and a memory outside an SPP core. For example, when data are transferred from an LM to the CSM_r, the DMAC reads data in the LM via the right on-chip bus, and the data are written on the CSM_r from the DMAC. We need two bus transactions for this data transfer. On the other hand, if a DTU in a CPU core on the left cluster is used for the same transfer, data are read from an LM by the DTU in the core, and the data are written on the CSM_l via the on-chip bus by the DTU. Only one transaction on the on-chip bus is necessary in this case, and the data transfer is more efficient compared with the case using the off-core DMAC. Although each CPU core in the left cluster has an individual FVC, the SPP cores in the right cluster share an FVC. With this simpler FVC configuration, all SPP cores operate at the same voltage and the same frequency, which are controlled simultaneously.

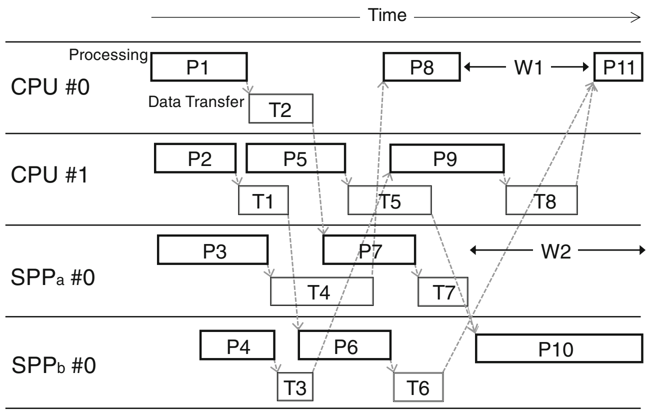


Fig. 2.4 Parallel operation

When a program is executed on a heterogeneous multicore, it is divided into small parts, and each is executed in parallel in the most suitable processor core, as shown in Fig. 2.4. Each core processes data on its LM or cache in a Pi period, and the DTU of a core simultaneously executes a memory–memory data transfer in a Ti period. For example, CPU #1 processes data on its LM at a P2 period, and its DTU transfers processed data from the LM of CPU #1 to the LM of SPP_b #0 at the T1 period. After the data transfer, SPP_b #0 starts to process data on its LM at a P6 period. CPU #1 also starts a P5 process that overlaps with the T1 period. In the parallel operation of Fig. 2.4, there is a time slot like W1 when the corresponding core CPU #0 does not need to process or transfer data from the core. During this time slot, the frequencies of the PU and DTU of CPU #0 can be slowed down or stopped, or their power supplies can be cut off by control of the connected FVC. As there are no internal operations of SPP_a #0 during the time slot W2, the power of SPP_a #0 can be cut off during this time slot. This FVC control reduces redundant power consumption of cores and can result in lowering the power consumption of a heterogeneous multicore chip.

Here, we show an example of our architecture model applied to a heterogeneous multicore chip. Figure 2.5 is a photograph of the RP-X chip (see Sect. 4.4) [3–5]. Figure 2.6 depicts the internal block diagram. The chip includes eight CPU cores and seven three-type SPP cores. The CPU (see Sect. 3.1) includes a two-level LM as well as a 32-KB instruction cache and a 32-KB operand cache. The LM consists of a 16-KB ILRAM for instruction storage, a 16-KB OLRAM for data storage, and a 64-KB URAM for instruction and data storage. Each CPU has a local clock pulse generator (LCPG) that corresponds to the FVC and controls the CPU’s clock frequency independently. The eight CPUs are divided into two clusters. Each cluster of four CPUs is connected to independent on-chip buses. Additionally, each cluster has a 256-KB CSM and a DDR3 port which is connected to off-chip DDR3 DRAMs.

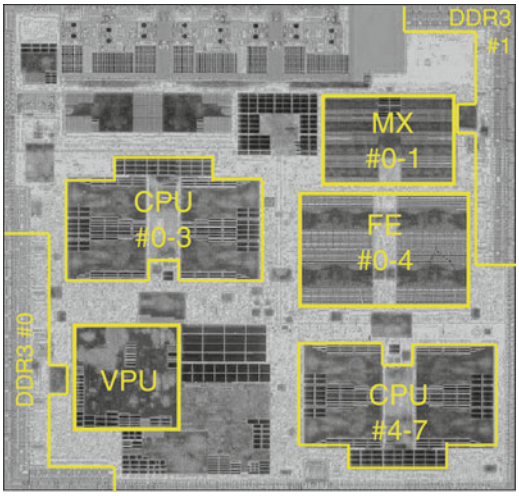


Fig. 2.5 Heterogeneous multicore chip

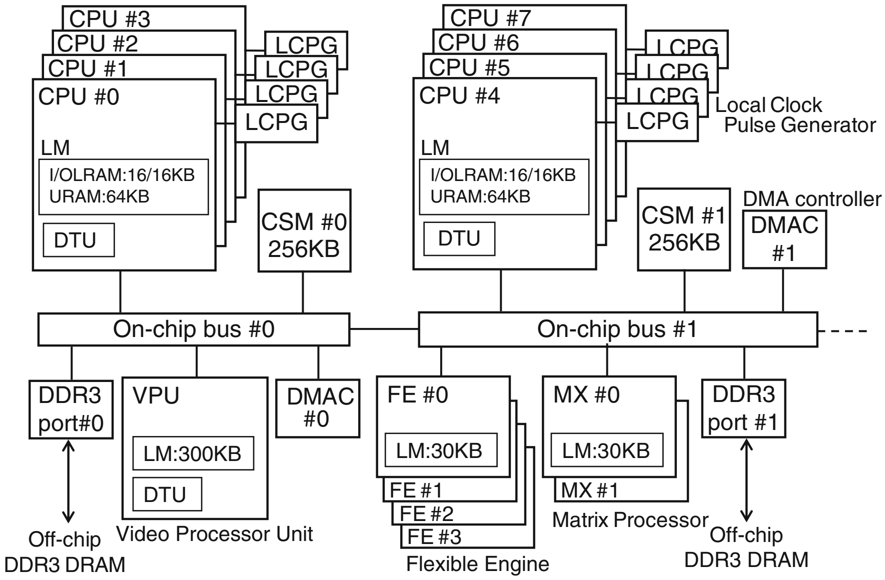


Fig. 2.6 Block diagram of heterogeneous multicore chip

Three types of SPPs are embedded on the chip. The first SPP is a video processing unit (VPU, see Sect. 3.4) which is specialized for video processing such as MPEG-4 and H.264 codec. The VPU has a 300-KB LM and a DTU built-in. The second and third SPPs are four flexible engines (FEs, see Sect. 3.2), and two matrix processors (MXs, see Sect. 3.3), and they are included in another cluster. The FE is a dynamically reconfigurable processor which is suitable for data-parallel processing such as digital signal processing. The FE has an internal 30-KB LM but does not have a DTU. The on-chip DMA controller (DMAC) that can be used in common by on-chip units or a DTU of another core is used to transfer data between the LM and other memories. The MX has 1,024-way single instruction multiple data (SIMD) architecture that is suitable for highly data-intensive processing such as video recognition. The MX has an internal 128-KB LM but does not have its DTU, just as with the FE. In the chip photograph in Fig. 2.5, the upper-left island includes four CPUs, and the lower-left island has the VPU with other blocks. The left cluster in Fig. 2.6 includes these left islands and a DDR3 port depicted at the lower-left side. The lower-right island in the photo in Fig. 2.5 includes another four CPUs, the center-right island has four FEs, and the upper-right has two MXs. The right cluster in Fig. 2.6 includes these right islands and a DDR3 port depicted at the upper-right side. With these 15 on-chip heterogeneous cores, the chip can execute a wide variety of multimedia and digital-convergence applications at high-speed and low-power consumption. The details of the chip and its applications are described in Chaps. 4–6.

2.2 Address Space

There are two types of address spaces defined for a heterogeneous multicore chip. One is a public address space where all major memory resources on and off the chip are mapped and can be accessed by processor cores and DMA controllers in common. The other is a private address space where the addresses looked for from inside the processor core are defined. The thread of a program on a processor core runs on the private address space of the processor core. The private address space of each processor core is defined independently.

Figure 2.7a shows a public address space of the heterogeneous multicore chip depicted in Fig. 2.1. The CSM, the LMs of CPU #0 to CPU #m, the LMs of SPP_a #0 to SPP_a #n, and the LMs of SPP_b #0 to SPP_b #k are mapped in the public address space, as well as the off-chip main memory. Each DTU in each processor core can access the off-chip main memory, the CSM, and the LMs in the public address space and can transfer data between various kinds of memories. A private address space is independently defined per processor core. The private addresses are generated by the PU of each processor core. For a CPU core, the address would be generated during the execution of a load or store instruction in the PU. Figure 2.7b, c shows examples of private address spaces of a CPU and SPP. The PU of the CPU core accesses data of the off-chip main memory, the CSM, and its own LM mapped on the private address space of Fig. 2.7b. If the LM of another processor core is not mapped on this private address space, the load/store instructions executed by the PU of the CPU core cannot access data on the other processor core's LM. Instead, the DTU of the CPU core transfers data from the other processor core's LM to its own LM, the CSM, or the off-chip main memory using the public address space, and the PU accesses the data in its private address space. In the SPP example (Fig. 2.7c), the PU of the SPP core can access only its own LM in this case. The data transfer

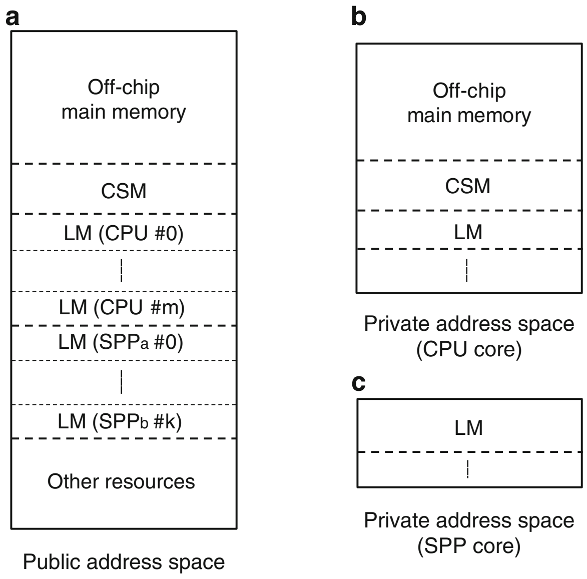


Fig. 2.7 Public/private address spaces

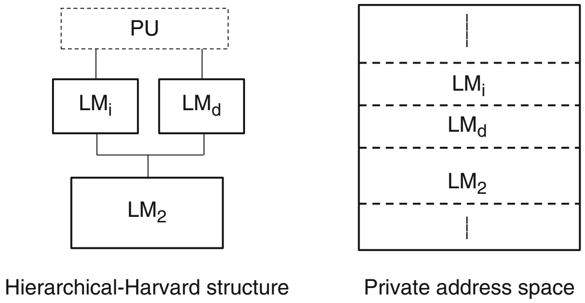


Fig. 2.8 Private address space (Hierarchical Harvard)

between its own LM and memories outside the core is done by its own DTU on the public address space.

The address mapping of a private address space varies according to the structure of the local memory. Figure 2.8 illustrates the case of the hierarchical Harvard structure of Fig. 2.2c. The LM_i and LM_d are first-level local memories for instructions and data, respectively. The LM_2 is a second-level local memory that stores both instructions and data. The LM_i, LM_d, and LM_2 are mapped on different address areas in the private address space. The PU accesses each LM with different addresses.

The size of the address spaces depends on the implementation of the heterogeneous multicore chip and its system. For example, a 40-bit address is assigned for a public address space, a 32-bit address for a CPU core's private address space, a 16-bit address for the SPP's private address space, and so on. In this case, the sizes of each space are 1 TB, 4 GB, and 64 KB, respectively. Concrete examples of this are described in Chaps. 3 and 4.

References

1. Uchiyama K (2008) Power-efficient heterogeneous parallelism for digital convergence, digest of technical papers of 2008 Symposium of VLSI circuits, Honolulu, USA, pp 6–9
2. Uchiyama K (2010) Power-efficient heterogeneous multicore for digital convergence, Proceedings of 10th International Forum on Embedded MPSoC and Multicore, Gifu, Japan, pp 339–356
3. Yuyama Y, et al (2010) A 45 nm 37.3GOPS/W heterogeneous multi-core SoC, ISSCC Dig: 100–101
4. Nito T, et al (2010) A 45 nm heterogeneous multi-core SoC supporting an over 32-bits physical address space for digital appliance, COOL Chips XIII Proceedings, Session XI, no. 1
5. Arakawa F (2011) Low power multicore for embedded systems, CMOS Emerging Technology 2011, Session 5B, no. 1