System-on-Chip (SoC) design

Lecture 08 on Dedicated systems

Teacher: Giuseppe Scollo

University of Catania
Department of Mathematics and Computer Science
Graduate Course in Computer Science, 2016-17

Table of Contents

outline:

- the system-on-chip (SoC) concept
  - interplay of the main components
  - SoC interfaces for custom hardware
- SoC design principles
  - heterogeneous and distributed data processing
  - heterogeneous and distributed communications
  - heterogeneous and distributed storage
  - hierarchical control
- example: a multi-media SoC
  - SoC architecture
  - design analysis

SoC: a domain-specialized *platform* on a single chip

- the application domain greatly affects the type of hardware peripherals, the size of memories and the nature of on-chip communications
- a SoC is a specialized yet *versatile* HW/SW system

domain examples:

mobile telephony, video processing, high-speed networking

video-processing application examples:

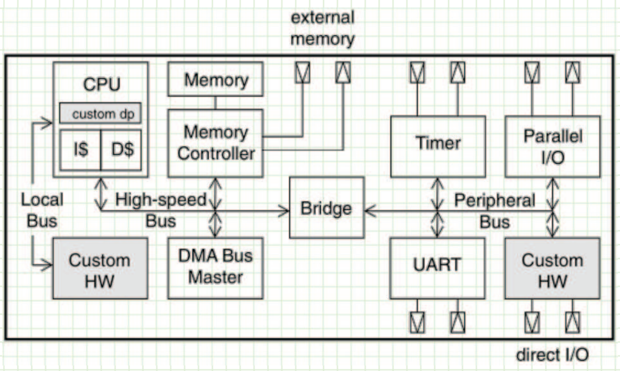
image transcoding, (de)compression, color transformations

advantages of domain specialization:

- *specialization* → higher processing efficiency
  → lower power consumption or higher processing speed
- *versatility* → reusability over multiple applications in the domain
  → lower design cost per application and cheaper SoC

interplay of the main components

four orthogonal dimensions of analysis of the organization and interplay of components in a SoC:
control, computation, communication, storage



Schaumont, Figure 8.1 - Generic template for a system-on-chip

in this generic architecture:

- top-level control by a general-purpose microprocessor (typically RISC), specific control functions by other components
- computation, communication, and storage, are both *heterogeneous* and *distributed*

SoC interfaces for custom hardware

shaded blocks in fig. 8.1 show three ways to attach custom hardware in a SoC context

- as a *standard peripheral* on a system bus
  most general approach: communication through R/W memory-mapped I/O
  (+) uniform, universal mechanism
  (-) poor scalability, system bus quickly becomes a bottleneck
- as a *coprocessor* through a local bus or coprocessor interface
  communication through a dedicated protocol
  (+) higher bandwidth, lower latency
  (-) dependency on the bus protocol or coprocessor interface provided by the microprocessor
- as a custom-hardware *datapath inside the microprocessor*
  extension of the instruction set, communication through the register file
  (+) very high bandwidth
  (-) strong dependency on the microprocessor and on its bottlenecks

in SoC design, there is no single *best* way to integrate hardware and software

design principles in SoC architecture

factors to trade-off in SoC design include:

- required communication bandwidth
- design complexity of the custom hardware interface
- software development
- available design time
- overall cost budget

four design principles for *any* SoC:

- heterogeneous and distributed communications
- heterogeneous and distributed data processing
- heterogeneous and distributed storage
- hierarchical control

heterogeneous and distributed data processing

hardware heterogeneity : FSMDs, microprogrammed engines, RISC microprocessors

all capable of word-level and instruction-level parallelism, yet none supports (true) task-level parallelism—they are sequential machines at this level

task-level parallelism is possible in a SoC thanks to their *multiplicity*

functional heterogeneity : computationally different units

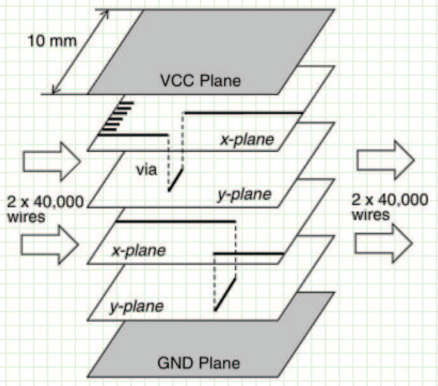
specialized units for different applications in the domain, e.g. image analysis, compression, processing in a DSP chip

thanks to parallelism at all levels a SoC can fully exploit the hardware technology, in two respects:

- the specialization of each computational unit allows its silicon implementation to be much more efficient than a software one running on a general.purpose RISC
- functionally concurrent units may execute truly parallel computations

multiple *bus segments* using bus bridges may prevent the central bus bottleneck

large variations of on-chip communication requirements call for *heterogeneity* of on-chip connections: shared busses, point-to-point, serial, parallel connections



Schaumont, Figure 8.2 - Demonstration of the routing density in a six-layer metal 90 nm CMOS chip

an example by Tensilica founder Chris Rowen about extremely high bandwidth of on-chip communication:

assumptions:

- 90nm six-layer metal processor, with two pairs of metal layers for wire routing
- wire density: 4/μm, bit frequency: 500 MHz

→ a theoretical bandwidth of 40 Tbps!

off-chip communication is orders of magnitude below

e.g. a four-port Hypertransport 3.1 interconnect, a standard for high-speed processors, gives a ~ 200 Gbps bandwidth

heterogeneity of silicon-based memories in a SoC is summarized in Table 8.1

Type	Register Register file	DRAM	SRAM	NVROM (ROM, PROM, EPROM)	NVRAM (Flash, EEPROM)
Cell size (bit)	10 transistors	1 transistor	4 transistors	1 transistor	1 transistor
Retention	0	Tens of ms	0	∞	10 years
Addressing	Implicit	Multiplexed	Non-muxed	Non-muxed	Non-muxed
Access time	Less then 1 ns	Less then 20 ns	Less then 10 ns	20 ns	20 ns (read) 100 μs (write)
Power consumption	High	Low	High	Very low	Very low
Write durability	∞	∞	∞	∞	One million times

Schaumont, Table 8.1 - Types of memories

distributed storage significantly complicates the concept of a centralized memory address space, when data need to be shared among components

- multiple copies of a single data item in different memories are to be kept consistent
- updating of a shared data item is to be implemented in a way that will not violate data dependencies among the components that share it (see Problem 8.1)

hierarchical control

a control hierarchy among components ensures that the entire SoC operates as a single logical entity

- the central control is usually played by a RISC processor, which sends commands to other components
- these may operate loosely, almost independently, from one another, but at some point they need to synchronize and report back to the central point of control

local control may be played by dedicated components, such as coprocessors or other custom hardware, but their operations and those of the central controller are not entirely independent

for example, they may synchronize, on the sending of data for required computations and receiving the result by the central controller, by means of a handshake protocol

the design of a good control hierarchy is a challenging problem

- it should exploit the distributed nature of the SoC as much as possible
- while it should minimize the number of conflicts that arise as a result of parallel execution

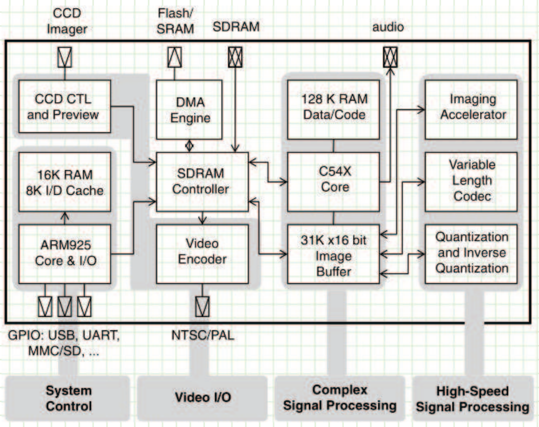
depending on the workload distribution, any component may cause a system bottleneck

the challenge for the SoC designer (or platform programmer) is to be aware of the location of such system bottlenecks, and to control them

example: a multimedia SoC

real case study: a digital media processor by Texas Instruments

manufactured in 130 nm CMOS, used for the processing of still images, video, and audio in portable, battery-operated devices, consumes 250—400 mW



Schaumont, Figure 8.3- Block diagram of portable multi-media system

several *device modes*, including:

- live preview of images (default)
- live-video compression (MPEG, MJPEG) and streaming into external memory
- high-resolution still-image capturing and JPEG conversion
- live audio capturing and compression to MP3, WMA, or AAC
- video decode and playback of a recorded stream onto a video display
- still image decode and playback of a stored image onto a video display
- audio decode and playback
- photo printing of a stored image into a format suitable for a photo printer

design analysis of the example SoC

four specialized subsystems are shaded in figure 8.3, centered around the SDRAM controller which organizes the traffic to the large, off-chip memory holding image data

the previously discussed four properties can be recognized in this chip:

- *heterogeneous and distributed processing*: hardwired (video subsystem), signal processing (DSP), general purpose (ARM processor)
- *heterogeneous and distributed communications*: no central bus system, rather a central controller that multiplexes access to off-chip memory, plus local bus systems between specialized processors and their local memories
- *heterogeneous and distributed storage*: large off-chip SDRAM, instruction memories attached to TI DSP and to ARM, dedicated data memories acting as local buffers
- *hierarchical control*: a hierarchy of control ensures optimality of the overall parallelism, where the ARM starts/stops components and controls data streams depending on the device mode

references

recommended readings:

Schaumont (2012) Ch. 8, Sect. 8.1-8.3

for further consultation:

Schaumont (2012) Ch. 8, Sect. 8.4
Rowen (2004) Ch. 1