Unit IV - System Design Techniques & Networks.

- Design methodology is employed in technology fields, including internet, software & information systems development.

Common Design Methodology:
1. Top down Design or Stepwise Refinement
2. Bottom up Design
3. Structured Design
4. Structured Analysis & Design Techniques
5. Data Structured Systems Development
6. Object Oriented Design

- Design Process is important when people work together for designing complex embedded systems.

Typical specifications for a product will include,
- Functionality
- Manufacturing cost
- Performance
- Power Consumption
- Design cost
- Quality
Design flows
- Design flow is a sequence of steps to be followed during a design.
- Some steps can be performed by tools, such as compilers or CAD systems.
- Other steps can be performed by hand.

- Waterfall model
  - introduced by Royce
  - First model proposed for software development process

  Requirements → Architecture

  5 Phases
  1. Requirement Analysis
     determines the basic characteristics of the system
  2. Architecture Design
     decomposes the functionality into major components
Coding - implements the pieces and integrates them.

Testing - exercises and uncovers bugs.

Maintenance - entails deployment in the field, fixes bugs and upgrades.
Problems:
1. Only local feedback - may need iteration between coding and requirements.
2. Doesn’t integrate top-down and bottom-up design.
3. With a limited amount of feedback to the next higher level of abstraction.
4. Unrealistic design process.

(ii) Spiral model - Alternate model for software development process.

- System feasibility
- Specification
- Prototype
- Initial system
- Enhanced system
- Several versions of the system will be built.
- More realistic than waterfall model.

(iii) Successive Refinement

Specify → Architect → Design → Build → Test

Specify → Architect → Design → Build → Test

Initial System → Refined System

Initial system - Rough prototype.
Successive models of the system - Refined.

(iv) Hardware/Software design

Requirement & Specification → Architecture

Hardware Design → Software Design

Integration → System Test.
Front end activities - specification & architecture.
Back end activities - integration & testing
In middle - development of hardware & software components.

(v) Hierarchical design flows:

- Requirements and specification
  - Architecture
    - Hardware design
    - Software design
    - Integration
    - System test

- Specification
  - Hardware architecture
    - Detailed design
    - Another design cycle
    - Least abstract
  - Software architecture
    - Module design
    - Another design cycle
    - Least abstract
  - Integration
    - Test
    - Moderately abstract

- Test
  - Least abstract
  - Moderately abstract
  - Most abstract
(vi) Concurrent Engineering,
- Attempts to take a broader approach and optimize the total flow.
- Reduced design time is an important goal.

Requirements Analysis
Def. of Requirement
- Informal descriptions of what the customer wants.

Def. of Specification
- More detailed, precise & consistent descriptions of the system that can be used to create the architecture.

Objective of Requirements
- Effective communication between the customer & the designers.

Types of Requirements
- Functional
- Non-functional (Physical size, cost, Power consumption, Design time, Reliability)
Tests for good set of requirements:
- Correctness
- Unambiguity
- Completeness
- Verifiability
- Consistency
- Modifiability
- Traceability

Setting requirements:
- Customer interviews
- Comparison with competitors
- Sales feedback
- Prototypes

Specifications:
Control Oriented Specification Language
State Machine Specification Language (or)
SDL language
- developed by communication industry
- for specifying communication protocols, telephone systems, and so on
- Event oriented state machine model
Symbols used:

- ○ State
- △ Input
- □ Output
- ▶ Task
- ◻ Decision
- ◼ Save

Graphical Specification:

1. Telephone on-hook
2. Caller goes off-hook
3. Caller gets dial tone

State Chart:

- State based specification
- Event-driven model
- States are grouped together to show common functionality
- Basic groupings - OR, AND

Traditional:

Machine goes to state s4 from any of s1, s2 or s3 when they receive input i2.
AND state in state charts.

- AND state $s_{ab}$ decomposed into components $s_a$ & $s_b$.
- When the machine enters the AND state, it simultaneously inhabits the state $s_1$ of component $s_a$ & the state $s_3$ of component $s_b$. 
System Analysis & Architecture Design
- Using CRC card methodology.

**CRC card**
- Well suited to object-oriented design.
- Supports encapsulation of data & functions.

C - Classes → logical groupings of data & functions
R - Responsibilities → what the classes do.
C - Collaborations → relationship with other classes.

**Layout of a CRC card**

Class Name:
Superclasses:
Subclasses:
Responsibilities:
Collaborators:

Class Name:
Class's function
Attributes:

**Advantages of CRC cards**
- Creating CRC cards is easy for non-computer people.
- Getting the advice of domain experts is important in system design.
Review of CRC

- Class is used to easily transformable into a class definition in an object-oriented design.

* Quality Assurance.

- A process is vital for the delivery of a satisfactory system.

Low Quality products - Reasons:

1. Poorly manufactured.
2. Components were improperly designed.
3. Product's requirements were poorly understood.

QA Techniques:

- International Standards Organization (ISO) has created a set of quality standards known as ISO 9000.
- ISO 9000 was created to apply to a broad range of industries.
- Processes used to satisfy ISO 9000 affect the entire organization as well as the individual steps taken during design and manufacturing.
Observations in 550 9000

- Documentation is important.
- Documentation helps internal quality monitoring groups to ensure that the required processes.
- Documentation helps outside groups to understand the processes and how they are being implemented.

Metrics are used in quality control process, to know whether, the levels of

Role of 550 9000

- To help the organizations to study their total process, not just particular segments.
- One way to measure the quality of Organization’s software development process is the

...
This CMM provides a benchmark by which organizations can judge themselves and use that information for improvement.

CMM defines 5 levels of maturity,

1. Initial - Poorly organized process.
2. Repeatable - Basic tracking mechanisms.
3. Defined - Documentation.
4. Managed - detailed measurement of development process.
5. Optimizing - highest level, feedback from detailed measurements.
Why distributed?

• Higher performance at lower cost.
• Physically distributed activities---time constants may not allow transmission to central site.
• Improved debugging---use one CPU in network to debug others.
• May buy subsystems that have embedded processors.
Network abstractions

• International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) model to describe networks:
  – 7-layer model.

• Provides a standard way to classify network components and operations.
OSI model

- **Physical layer**
  - mechanical, electrical

- **Data link layer**
  - reliable data transport

- **Network layer**
  - end-to-end service

- **Transport layer**
  - connections

- **Session layer**
  - application dialog control

- **Presentation layer**
  - data format

- **Application layer**
  - end-use interface

**Diagram:**

- application
- presentation
- session
- transport
- network
- data link
- physical
OSI layers

- **Physical**: connectors, bit formats, etc.
- **Data link**: error detection and control across a single link (single hop).
- **Network**: end-to-end multi-hop data communication.
- **Transport**: provides connections; may optimize network resources.
OSI layers, cont’d.

- **Session**: services for end-user applications: data grouping, checkpointing, etc.
- **Presentation**: data formats, transformation services.
- **Application**: interface between network and end-user programs.
Vehicles as networks

• 1/3 of cost of car/airplane is electronics/avionics.
• Dozens of microprocessors are used throughout the vehicle.
• Network applications:
  – Vehicle control.
  – Instrumentation.
  – Communication.
  – Passenger entertainment systems.
CAN bus

- First used in 1991.
- Serial bus, 1 Mb/sec up to 40 m.
- Synchronous bus.
- Logic 0 dominates logic 1 on bus.
- Arbitrated with CSMA/AMP:
  - Arbitration on message priority.
CAN data frame

- 11 bit destination address.
- RTR bit determines read/write from/to destination.
- Any node can detect bus error, interrupt packet for retransmission.
CAN controller

• Controller implements physical and data link layers.

• No network layer needed---bus provides end-to-end connections.
Other vehicle busses

• FlexRay is next generation:
  – Time triggered protocol.
  – 10 Mb/s.

• Local Interconnect Network (LIN) connects devices in a small area (e.g., door).

• Passenger entertainment networks:
  – Bluetooth.
  – Media Oriented Systems Transport (MOST).
Automobile network

- Engine provides power to drive the wheels via the transmission.
- Transmission adjusts gearing based on operating characteristics.
- ABS controls brakes.
Freescale MPC5676R

- Designed for cars and power trains.
- Two Power cores with vector units.
- Time processing units can sense and generate waveforms, offloading CPUs.
Avionics

- Aviation electronics must be certified.
- Early systems used line replaceable units which operated separately.
- Boeing 777 uses a bus-based system with core processor modules connected by SAFEbus.
- Federated network has networks for functions.
- Boeing 787 does not fix the mapping from applications to network units.
I²C bus

• Designed for low-cost, medium data rate applications.

• Characteristics:
  – serial;
  – multiple-master;
  – fixed-priority arbitration.

• Several microcontrollers come with built-in I²C controllers.
\( \text{I}^2\text{C physical layer} \)

Diagram showing the physical layer of an \( \text{I}^2\text{C} \) interface, with master 1 and master 2 connected to slave 1 and slave 2 via a data line and clock line.
I²C data format

SCL

SDL

start  MSB  ack
I²C electrical interface

- Open collector interface:
I²C signaling

• Sender pulls down bus for 0.
• Sender listens to bus---if it tried to send a 1 and heard a 0, someone else is simultaneously transmitting.
• Transmissions occur in 8-bit bytes.
I²C data link layer

• Every device has an address (7 bits in standard, 10 bits in extension).
  – Bit 8 of address signals read or write.

• General call address allows broadcast.
I²C bus arbitration

• Sender listens while sending address.
• When sender hears a conflict, if its address is higher, it stops signaling.
• Low-priority senders relinquish control early enough in clock cycle to allow bit to be transmitted reliably.
I²C transmissions

multi-byte write

read from slave

write, then read
Ethernet

- Dominant non-telephone LAN.
- Versions: 10 Mb/s, 100 Mb/s, 1 Gb/s
- Goal: reliable communication over an unreliable medium.
Ethernet topology

- Bus-based system, several possible physical layers:

A B C
CSMA/CD

• Carrier sense multiple access with collision detection:
  – sense collisions;
  – exponentially back off in time;
  – retransmit.
Exponential back-off times
Ethernet packet format

- preamble
- start frame
- source adrs
- dest adrs
- length
- data payload
- padding
- CRC
Ethernet performance

• Quality-of-service tends to non-linearly decrease at high load levels.
• Can’t guarantee real-time deadlines. However, may provide very good service at proper load levels.
Fieldbus

• Used for industrial control and instrumentation---factories, etc.
• H1 standard based on 31.25 MB/s twisted pair medium.
• High Speed Ethernet (HSE) standard based on 100 Mb/s Ethernet.
MPSOCS and Shared Memory Multiprocessors
Networks and multiprocessors

• Single-chip multiprocessors.
• Accelerators and heterogeneous multiprocessors.
Shared memory multiprocessors

• Multi-core systems are symmetric multiprocessors with identical PEs.
• Many embedded multiprocessors are heterogeneous:
  – Multiple programmable CPU types.
  – Hardwired accelerators.
• Heterogeneous multiprocessors use less energy, are less expensive.
TI TMS320DM816x DaVinci

- ARM Cortex A8 plus C674x VLIW DSP.
- HD video coprocessor accelerates H.264, MPEG-4, etc.
- HD video processing subsystem provides additional video processing.
- Graphics unit performs 3D graphics.
Accelerated systems

- Use additional computational unit dedicated to some functions?
  - Hardwired logic.
  - Extra CPU.

- **Hardware/software co-design**: joint design of hardware and software architectures.
Accelerated system architecture
Accelerator vs. co-processor

• A co-processor executes instructions.
  – Instructions are dispatched by the CPU.

• An accelerator appears as a device on the bus.
  – The accelerator is controlled by registers.
Accelerator implementations

• Application-specific integrated circuit.
• Field-programmable gate array (FPGA).
• Standard component.
  – Example: graphics processor.
Xilinx Zynq-7000

- Dual-CPU ARM MP Core augmented with an FPGA fabric.
- AMBA bus connects to CPUs and FPGA fabric.
System design tasks

• Design a heterogeneous multiprocessor architecture.
  – Processing element (PE): CPU, accelerator, etc.

• Program the system.
Accelerated system design

• First, determine that the system really needs to be accelerated.
  – How much faster is the accelerator on the core function?
  – How much data transfer overhead?

• Design the accelerator itself.

• Design CPU interface to accelerator.
Accelerated system platforms

• Several off-the-shelf boards are available for acceleration in PCs:
  – FPGA-based core;
  – PC bus interface.
Accelerator/CPU interface

- Accelerator registers provide control registers for CPU.
- Data registers can be used for small data objects.
- Accelerator may include special-purpose read/write logic.
  - Especially valuable for large data transfers.
System integration and debugging

• Try to debug the CPU/accelerator interface separately from the accelerator core.
• Build scaffolding to test the accelerator.
• Hardware/software co-simulation can be useful.
Caching problems

• Main memory provides the primary data transfer mechanism to the accelerator.

• Programs must ensure that caching does not invalidate main memory data.
  – CPU reads location S.
  – Accelerator writes location S.
  – CPU writes location S.
Synchronization

• As with cache, main memory writes to shared memory may cause invalidation:
  – CPU reads S.
  – Accelerator writes S.
  – CPU reads S.
Multiprocessor performance analysis

• Effects of parallelism (and lack of it):
  – Processes.
  – CPU and bus.
  – Multiple processors.
Accelerator speedup

• Critical parameter is speedup: how much faster is the system with the accelerator?
• Must take into account:
  – Accelerator execution time.
  – Data transfer time.
  – Synchronization with the master CPU.
Accelerator execution time

- Total accelerator execution time:
  \[ t_{\text{accel}} = t_{\text{in}} + t_x + t_{\text{out}} \]
Accelerator speedup

• Assume loop is executed n times.
• Compare accelerated system to non-accelerated system:
  \[ S = n(t_{CPU} - t_{accel}) \]
  \[ = n[t_{CPU} - (t_{in} + t_{x} + t_{out})] \]

Execution time on CPU
Single- vs. multi-threaded

• One critical factor is available parallelism:
  – single-threaded/blocking: CPU waits for accelerator;
  – multithreaded/non-blocking: CPU continues to execute along with accelerator.

• To multithread, CPU must have useful work to do.
  – But software must also support multithreading.
Total execution time

- Single-threaded:

- Multi-threaded:
Execution time analysis

• Single-threaded:
  – Count execution time of all component processes.

• Multi-threaded:
  – Find longest path through execution.
Sources of parallelism

• Overlap I/O and accelerator computation.
  – Perform operations in batches, read in second batch of data while computing on first batch.

• Find other work to do on the CPU.
  – May reschedule operations to move work after accelerator initiation.
Data input/output times

• Bus transactions include:
  – flushing register/cache values to main memory;
  – time required for CPU to set up transaction;
  – overhead of data transfers by bus packets, handshaking, etc.
Scheduling and allocation

• Must:
  – schedule operations in time;
  – allocate computations to processing elements.

• Scheduling and allocation interact, but separating them helps.
  – Alternatively allocate, then schedule.
Example: scheduling and allocation

Task graph

Hardware platform

P1 → P3

P2 → P3

d1
d2

M1
M2
First design

- Allocate P1, P2 -> M1; P3 -> M2.
Second design

- Allocate P1 -> M1; P2, P3 -> M2:
Example: adjusting messages to reduce delay

- Task graph:

- Network:

  Transmission time = 4
Initial schedule

M1
P1

M2
P2

M3

P3

network
d1
d2

Time = 15
New design

• Modify P3:
  – reads one packet of d1, one packet of d2
  – computes partial result
  – continues to next packet
New schedule

M1
- P1

M2
- P2

M3
- P3
- d1
- d2
- d1
- d2
- d1
- d2
- d1
- d2

Network

Time = 12

Time
Buffering and performance

• Buffering may sequentialize operations.
  – Next process must wait for data to enter buffer before it can continue.
• Buffer policy (queue, RAM) affects available parallelism.
Buffers and latency

- Three processes separated by buffers:
Buffers and latency schedules


Must wait for all of A before getting any B