ECEN 478: Senior Design

Dr. Mahmoud Nabil
mnmahmoud@ncat.edu

North Carolina A & T State University

February 18, 2022

Outline

Motivation

Team of engineers who build a system need:

- An abstraction of the system
- An unambiguous communication medium
- A way to describe the subsystems
  - Inputs
  - Outputs
  - Behavior
- Functional Decomposition
  - Function – transformation from inputs to outputs
  - Decomposition – reduce to constituent parts

Learning Objectives

By the end of this chapter, you should:

- Understand the differences between bottom-up and top-down design.
- Know what functional decomposition is and how to apply it.
- Be able to apply functional decomposition to different problem domains.
- Understand the concept of coupling and cohesion, and how they impact design.

Bottom Up

- Given constituent parts
- Develop a working system
  - Build modules to accomplish specific tasks
  - Integrate modules together into working system
- For example
  - Given a supply AND, OR and NOT gates.
  - Build a computer
- Pros
  - Leads to efficient subsystem
- Cons
  - Complexity is difficult to manage
  - Little thought to designing reusable modules
  - Redesign cycles

Top Down

- Given the specification of a system
- Develop a working system
  - Divide the problem into abstract modules
  - Reiterate until constituent parts are reached
- Pros
  - Highly predictable design cycle
  - Efficient division of labor
- Cons
  - More time spent in planning

Outline

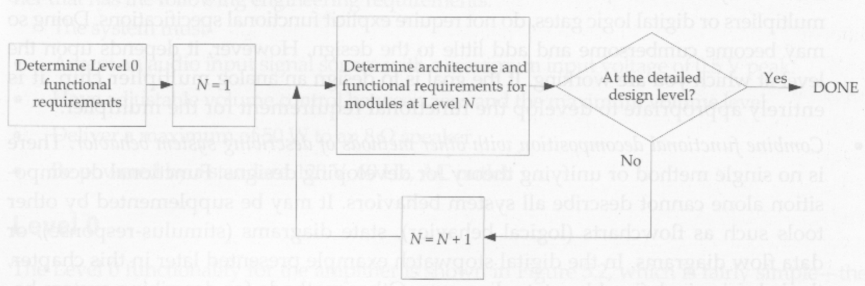
Functional Decomposition

It is the process of identifying the input, output, and the components needed to perform the functionality.

- It has a top-dow flavor
- Detailed design is obtained.
- Interfacing between modules are also obtained.

Functional Decomposition Process

- Recursively divide and conquer
  - Split a module into several submodules
  - Define the input, output, and behavior
  - Stop when you reach realizable components



Guidance on Design

- The design process is iterative
- Upfront time saves redesign time later
- Submodules at the same level should have similar complexity
- Precise input, output, and behavior specifications
- Look to be creative

Guidance on Design

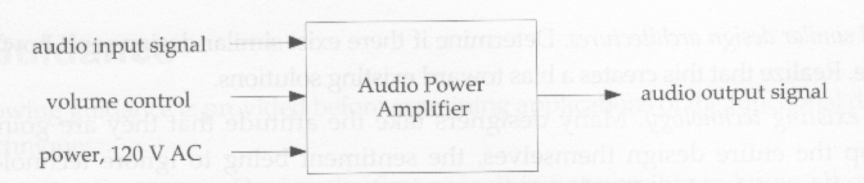
- Look at how it has been done before
- Use existing technology (to implement a submodule when you decide not to decompose it – i.e. combine top-down with bottom-up)
- Keep it simple
- Communicate results
  - With each other
  - With advisor
  - With instructor

Outline

Engineering requirements

- Accept an audio input signal source with a maximum input Voltage of 0.5 V peak.
- Have adjustable volume control between zero and the maximum volume level.
- Deliver a maximum of 50 W to an 8 Ohm speaker.
- Be powered by a standard 120 V, 60 Hz, AC outlet.

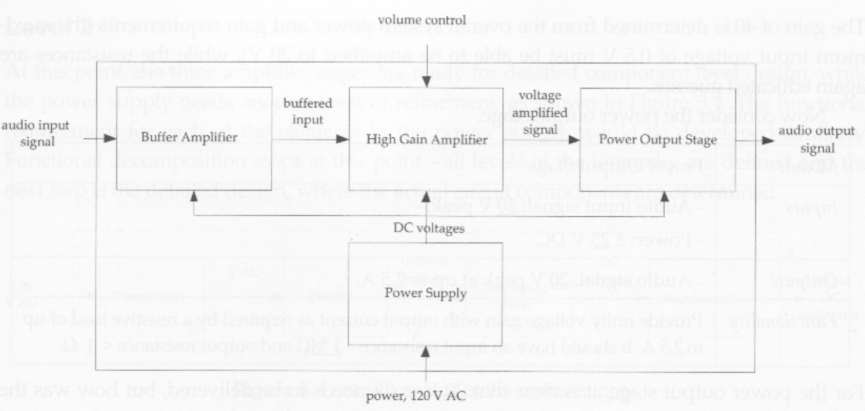
Level 0: Design



Level 0: Functional Requirement Table

Module	Audio Power Amplifier
Inputs	- Audio input signal: 0.5 V peak. - Power: 120 V AC rms, 60 Hz. - User volume control: variable control.
Outputs	- Audio output signal: ? V peak value.
Functionality	Amplify the input signal to produce a 50-W maximum output signal. The amplification should have variable user control. The output volume should be variable between no volume and a maximum volume level.

Level 1: Design



Buffer Amplifier: Functional Requirement Table

Module	Buffer amplifier
Inputs	- Audio input signal: 0.5 V peak. - Power: ± 25 V DC.
Outputs	- Audio signal: 0.5 V peak.
Functionality	Buffer the input signal and provide unity voltage gain. It should have an input resistance > 1 MΩ and an output resistance < 100 Ω.

High-gain Amplifier: : Functional Requirement Table

Module	High-gain amplifier
Inputs	- Audio input signal: 0.5 V peak. - User volume control: variable control. - Power: ± 25 V DC
Outputs	- Audio signal: 20 V peak.
Functionality	Provide an adjustable voltage gain, between 1 and 40. It should have an input resistance > 100 kΩ and an output resistance < 100 Ω.

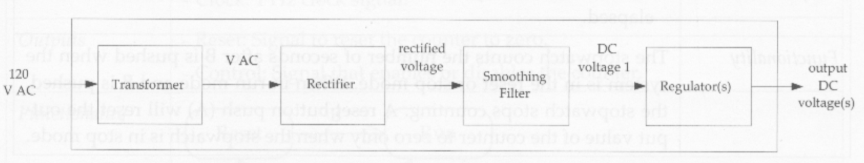
Power Output Stage: Functional Requirement Table

Module	Power Output Stage
Inputs	- Audio input signal: 20 V peak. - Power: ± 25 V DC.
Outputs	- Audio signal: 20 V peak at up to 2.5 A.
Functionality	Provide unity voltage gain with output current as required by a resistive load of up to 2.5 A. It should have an input resistance > 1 MΩ and output resistance < 1 Ω.

Power Supply: Functional Requirement Table

Module	Power Supply
Inputs	- 120 V AC rms.
Outputs	- Power: ± <u>25</u> V DC with up to <u>3.0</u> A of current with a regulation of < <u>1</u>%.
Functionality	Convert AC wall outlet voltage to positive and negative DC output voltages, and provide enough current to drive all amplifiers.

Level 2: Design (Power Supply)

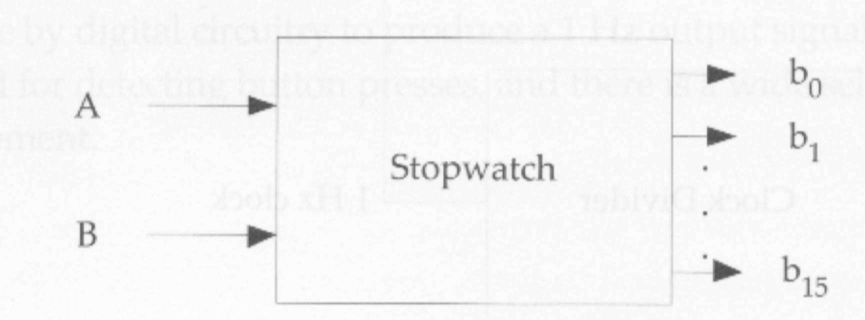


Outline

Engineering requirements

- Have no more than two control buttons.
- Implement run, stop, and reset functions
- Output 16-bit binary number that represents the number of seconds elapsed.

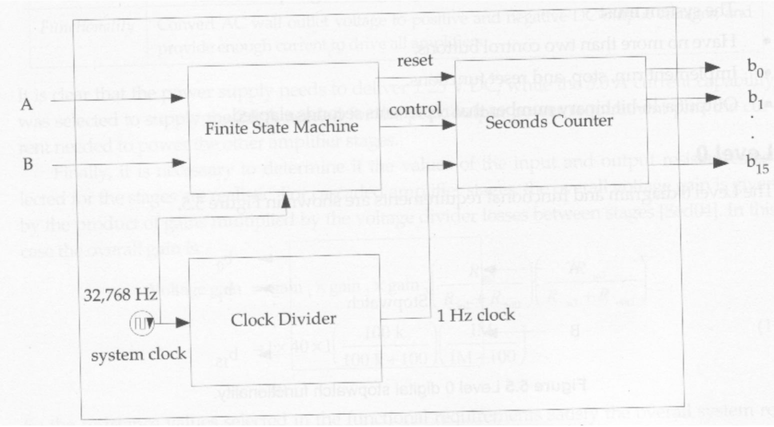
Level 0: Design



Stopwatch: Functional Requirement Table

Module	Stopwatch
Inputs	- A: Reset button signal. When the button is pushed it resets the counter to zero. - B: Run/stop toggle signal. When the button is pushed it toggles between run and stop modes.
Outputs	- b_{15}–b_0: 16-bit binary number that represents the number of seconds elapsed.
Functionality	The stopwatch counts the number of seconds after B is pushed when the system is in the reset or stop mode. When in run mode and B is pushed, the stopwatch stops counting. A reset button push (A) will reset the output value of the counter to zero only when the stopwatch is in stop mode.

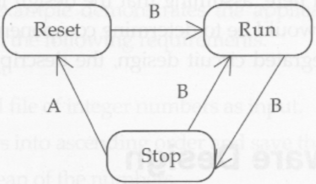
Level 1: Design



Seconds Counter: Functional Requirement Table

Module	Seconds Counter
Inputs	- Reset: Reset the counter to zero. - Control: Enable/disable the counter. - Clock: Increment the counter.
Outputs	- b_{15}–b_0: 16-bit binary representation of number of seconds elapsed.
Functionality	Count the seconds when enabled and resets to zero when reset signal enabled.

Finite State Machine: Functional Requirement Table

Module	Finite State Machine
Inputs	- A: Signal to reset the counter. - B: Signal to toggle the stopwatch between run and stop modes. - Clock: 1 Hz clock signal.
Outputs	- Reset: Signal to reset the counter to zero. - Control: Signal that enables or disables the counter.
Functionality	

Clock Divider: Functional Requirement Table

Module	Clock Divider
Inputs	- System clock: 32,768 Hz.
Outputs	- Internal clock: 1 Hz clock for seconds elapsed.
Functionality	Divide the system clock by 32,768 to produce a 1 Hz clock.

Outline

Types of Modules in Software Development

Structure Charts are specialized block diagrams for visualizing functional software design.

There are five types of modules in software design

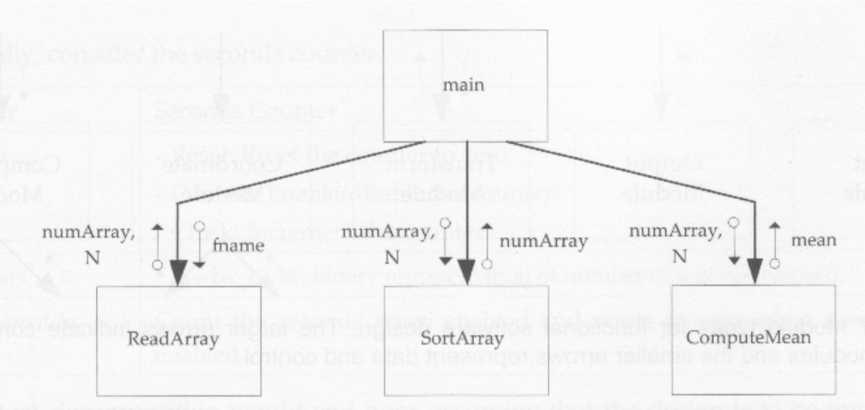
- Input Module
- Output Module
- Transform Module
- Coordinate Module
- Composite Module

Engineering Requirements

The system must

- Accept an ASCII file of integer numbers as input.
- Sort the numbers into ascending order and save the sorted numbers to disk.
- Compute the mean of the numbers.
- Display the mean on the screen.

Level 0: Design



Main Module: Functional Requirement Table

Module name	main()
Module type	Coordination
Input arguments	None.
Output arguments	None.
Description	The main function calls ReadArray() to read the input file from disk, SortArray() to sort the array, and ComputeMean() to determine the mean value of elements in the array. User interaction requires the user to enter the filename, and the mean value is displayed on the screen.
Modules invoked	ReadArray, SortArray, and ComputeMean.

Read Array: Functional Requirement Table

Module name	ReadArray()
Module type	Input and output
Input arguments	- fname[]: character array with filename to read from.
Output Arguments	- numArray[]: integer array with elements read from file. - N: number of elements in numArray[].
Description	Read data from input data file and store elements in array numArray[]. The number of elements read is placed in N.
Modules invoked	None.

Compute Mean: Functional Requirement Table

Module name	ComputeMean()
Module type	Input and output
Input arguments	- numArray[]: integer array of numbers. - N: number of elements in numArray[].
Output arguments	- mean: mean value of the elements in the array.
Description	Computes the mean value of the integer elements in the array.
Modules invoked	None.

Sort Array: Functional Requirement Table

Module name	SortArray()
Module type	Transformation
Input arguments	- numArray[]: integer array of numbers. - N: number of elements in numArray[].
Output Arguments	- numArray[]: sorted array of integer numbers.
Description	Sort elements in array using a shell sort algorithm. Saves the sorted array to disk.
Modules invoked	None.

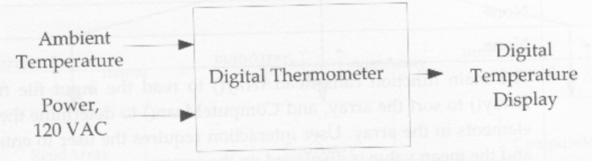
Outline

Engineering Requirements

The system must

- Measure temperature between 0 and 200 celesius
- Have an accuracy of 0.4% on the full scale
- Display the temperature digitally, including one digit beyond the decimal point.
- Be powered by a standard 120 V, 60 Hz AC outlet.

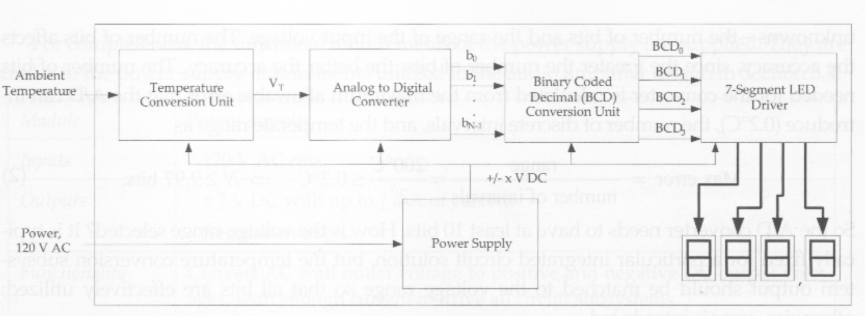
Level 0: Design



Digital Thermometer: Functional Requirement Table

Module	Digital Thermometer
Inputs	- Ambient temperature: 0–200°C. - Power: 120 V AC power.
Outputs	- Digital temperature display: A four digit display, including one digit beyond the decimal point.
Functionality	Displays temperature on digital readout with an accuracy of 0.4% of full scale.

Level 1: Design



Temperature Conversion Unit: Functional Requirement Table

Module	Temperature Conversion Unit
Inputs	- Ambient temperature: 0–200°C. - Power: ? V DC (to power the electronics).
Outputs	- V_T: temperature proportional voltage. $V_T = \alpha T$, and ranges from ? to ? V.
Functionality	Produces an output voltage that is linearly proportional to temperature. It must achieve an accuracy of ?%.

A/D Converter: Functional Requirement Table

Module	A/D Converter
Inputs	- V_T: voltage proportional to temperature that ranges from ? to ? V. - Power: ? V DC.
Outputs	- b_{N-1}–b_0: ?-bit binary representation of V_T.
Functionality	Converts analog input to binary digital output.

BCD Conversion Unit: Functional Requirement Table

Module	BCD Conversion Unit
Inputs	- 10-bit binary number (b_9–b_0): Represents the range 0.0–200.0°C. - Power: ? V DC.
Outputs	- BCD_0: 4-bit BCD representation of tenths digit (after decimal). - BCD_1: 4-bit BCD representation of ones digit. - BCD_2: 4-bit BCD representation of tens digit. - BCD_3: 4-bit BCD representation of hundreds digit.
Functionality	Converts the 10-bit binary number to BCD representation of temperature. Must refresh the displays twice a second.

Seven Segment: Functional Requirement Table

Module	Seven-Segment LED Driver
Inputs	- BCD_0: 4-bit BCD representation of tenths digit (after decimal). - BCD_1: 4-bit BCD representation of ones digit. - BCD_2: 4-bit BCD representation of tens digit. - BCD_3: 4-bit BCD representation of hundreds digit. - Power: ? V DC.
Outputs	- Four 7-segment driver lines.
Functionality	Converts the BCD for each digit into outputs that turn on LEDs in seven-segment package to display the temperature.

Power Supply: Functional Requirement Table

Module	Power supply
Inputs	- 120 V AC rms.
Outputs	- ± ? V DC with up to ? mA of current. - Regulation of ?%.
Functionality	Convert AC wall outlet voltage to positive and negative DC output voltages, with enough current to drive all circuit subsystems.

Outline

Introduction

We want to design components that are self-contained: independent, and with a single, well-defined purpose

Coupling

Coupling is the extent to which modules or subsystems are connected (i.e., low coupling means modules are as independent as possible from other modules, so that changes to module don't heavily impact other modules.)

- Phenomena of highly coupled systems
  - A failure in 1 module propagates
  - Difficult to redesign 1 module
- Phenomena of low coupled systems
  - Discourages reutilization of a module

Cohesion

Cohesion refers to how focused a module is (i.e., Cohesion often refers to how the elements of a module belong together.)

- Phenomena of highly cohesive systems
    - Easy to test modules independently
    - Simple (non-existent) control interface
- Phenomena of low cohesive systems
    - Less reuse of modules

Project Application: The Functional Design

- Design Level 0 User Interfacez
  - Present a single module block diagram with inputs and outputs identified.
  - Present the functional requirements: inputs, outputs, and functionality.
- Design Level 1
  - Present the Level 1 diagram (system architecture) with all modules and interconnections shown.
  - Describe the theory of operation. This should explain how the modules work together to achieve the functional objectives.
  - Present the functional requirements for each module at this level.
- Design Level N (for N>1)
  - Repeat the process from design Level 1 as necessary.

References

- Ford, Ralph Michael Coulston, Chris S - Design for electrical and computer engineers theory, concepts, and practice-McGraw-Hill (2008)

Questions