Embedded Systems

Ch 1. Introduction to Embedded Systems

Part A

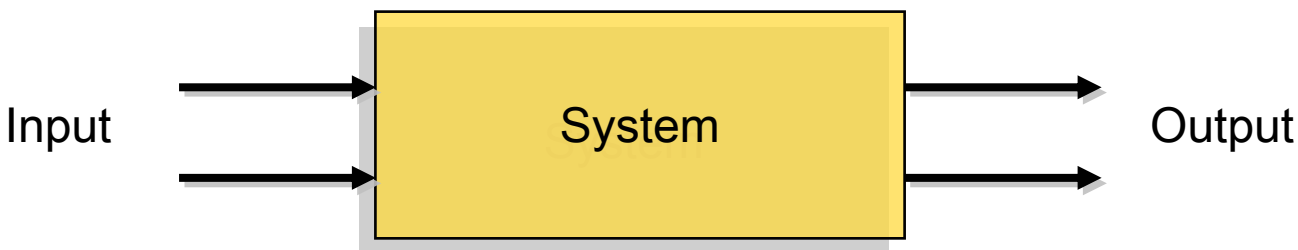
Byung Kook Kim
Dept of EECS
Korea Advanced Institute of Science and Technology

Overview

- 1.1 임베디드 시스템 정의
  - What is the embedded system?
- 1.2 임베디드 시스템 특징
  - Characteristics of embedded system
- 1.3 임베디드 시스템 응용제품
  - Embedded system products
- 1.4 임베디드 시스템 개발동향
  - Development trends of embedded system
- 1.5 Processor Architecture and Organization
- 1.6 Abstraction in Hardware Design

1.1 임베디드 시스템 정의

- 시스템 **(System)**
  - 내부는 알려지지 아니한 **black box**
  - 안으로 들어가는 입력과 밖으로 나오는 출력이 정의됨
  - 입력과 출력의 관계로 시스템의 동작을 기술함



임베디드 시스템 정의 (II)

- 실시간 시스템 **(Real-time system)**
  - 지정된 시간 내의 응답이 요구되는 시스템
  - **Hard real-time system**
    - 응답시간 이후의 결과는 치명적인 시스템
    - 예: 미시일 제어 장치, 비행기 항법 장치
  - **Soft real-time system**
    - 응답시간에 만족하지 않더라도 치명적인 결함을 가져오지는 않는 시스템
    - 예: **bank transfer**, 정보화 기기, 정보 가전

임베디드 시스템 정의 (III)

- 임베디드 시스템 (Embedded system, 내장형 시스템)
  - 특정 목적을 수행하기 위해 설계된 기기를 제어하기 위한 hardware와 software가 내장된 전자 응용 시스템
    - Electronic systems including hardware and software to control systems designed for specific purposes
  - Remarks
    - Microprocessor or microcontroller based system
    - Hardware 및 software의 유기적 결합을 통한 기능 구현
    - PC is a general-purpose system.

1.2 임베디드 시스템 특징

- 전문성과 범용성이 결합된 다양한 기능을 수행하는 제어 장치
  - 시스템의 사용목적이 제한되어있다
    - 자동차의 **ECU (Engine Control Unit)**
      - 엔진의 효율높은 제어
      - 엔진의 점화 타이밍, 밸브 등의 조작을 제어
    - 전자레인지 제어시스템
      - 최적의 음식을 조리하기 위해 조리 온도, 시간 등을 제어
  - 하드웨어 기술의 발달
    - 무어의 법칙: 컴퓨터 하드웨어는 **3**년에 **4**배의 속도로 집적도가 증가
  - 복합 고도 기능화
    - 제한된 임무를 보다 효율적이고 유연성 있게 수행
    - 내부 소프트웨어의 갱신 및 데이터의 저장
    - 휴대폰: 개인정보 입력, 일정 관리, 인터넷

임베디드 시스템 특징 (II)

- 산업제어장비로부터 네트워크 장비에 이르는 모든 분야에 활용되는 범용화된 기술
  - 초기: **4/8/16 bit microprocessor**
    - **Assembly language, C language**
    - 순차적으로 동작하는 응용 소프트웨어
    - **Calculator**
  - **32/64 bit microprocessor**
  - 메모리 가격의 하락
  - **Soc (System on a Chip)** 기술의 발달
    - 계산능력의 강화, 네트워킹, 멀티태스킹
    - 운영체제 **(operating system)**를 기반으로 하는 임베디드 시스템 구성

임베디드 시스템 특징 (III)

- 정보화 시대에서 모바일 컴퓨팅을 위한 기본기술
  - **Hardware** 구동용 **platform software**에서 응용 **software**로 기술 및 제품의 비중이 증가
    - 사용 가능한 운영체제의 일반화
    - 제품개발자: 응용 프로그램에 더 많은 시간을 할애하여 제품개발시간 단축 및 부가가치 증대
  - **Hardware**의 발전
    - **SoC (System on a chip)**기술의 발전
    - 주변기기 및 **memory** 등이 **microprocessor**에 내장 됨

1.3 Embedded System 응용 제품

- 산업용 장비
  - 가장 전통적인 **embedded system**
  - **Mini-computer**를 이용한 **digital control** 분야의 발달
  - 우주항공기술의 발달
  - 자동차
    - **BMW-7 series: 65 microprocessors. CAN (Controller Area Network)**
    - **Benz S class: 63 microprocessors**
  - 공장자동화
    - **PLC (Programmable Logic Controller) 1974**
      - 가장 많이 사용되는 산업용 제어장치
      - **Network** 기능의 추가
    - 공장 자동제어 시스템
      - 생산성 증대: 인건비 감소, 오류 감소, 품질의 균일화, 생산기간 단축
    - 자동화 조립 라인
      - 로봇, **conveyor belt**

Embedded System 응용제품 (II)

- 서비스 로봇
  - **Humanoid robot**
    - 인간을 닮은 사지: 보행, 대화 기능
  - 청소 로봇
    - 가시를 대신하는 로봇
  - 장애인 보조 로봇
    - 장애인의 보조. 삶의 질 향상
  - 중요기술
    - **Microprocessor** 이용
    - 인공지능 등의 알고리즘을 내장한 임베디드 시스템 기술
- 기타
  - 의학 기기
    - 내시경. 수술 로봇
  - 교통제어 기기
    - **Intelligent Transportation System**

Embedded System 응용제품 (III)

- 정보화 기기
  - 휴대폰
  - **PDA**
    - 휴대형 개인정보 단말기
    - **LCD display, touch screen**
    - 일정관리, 주소록, 메모장, 무선 **network**
  - **Home PDA**
    - 집안의 온도, 습도, 조도 등의 점검 및 제어
  - 스마트 폰
  - **Handheld PC**
  - **IC card**

Embedded System 응용 제품 (IV)

- 정보가전기기
  - 유무선 가정 **network**
  - 인공지능 스탠드
    - 주위의 밝기에 따라 조도 조절
  - 인터넷 냉장고
    - **Network** 기능을 이용한 정보의 검색 및 표시
  - 디지털 **TV**
    - **Set-top box**
  - 전자 레인지, 전기 밥솥, 진공청소기, 에어컨, 세탁기 등

Embedded System 응용 제품 (V)

- 네트워크 장비
  - Set-top box
    - 가정용 정보 network gateway
    - Embedded Linux 활용
  - Local Area Network (LAN) 장비
    - Switch
    - Hub
    - NIC: network Interface Controller
  - CDMA 및 무선화 기술의 발전
    - 새로운 network 시장의 창출

1.4 임베디드 시스템 개발동향

- Computing 기술의 발전
  - 70년대: host comruting
    - Unix host programming 및 수행. Host와 terminals
  - 90년대: client server
    - Server 중심. Client의 서비스 요청. World Wide Web
  - 00년대: Network computing
    - Mobile, Ubiquitous computing. Post-PC
    - Embedded 시스템 기술의 보편화

- Hardware 측면
  - SoC 기술이 주된 기술
    - 소형화, 전력소모 감소, 신뢰성 확보
    - 예: S3C4530 network processor

임베디드 시스템 개발동향 (II)

- Software 측면
  - Cross development environment (교차 개발 환경)
    - Host: PC. Windows or Linux
      - Cross compiler
    - Target: Embedded microprocessor
      - Downloading and execution
  - Embedded operating system (임베디드 운영 체제)
  - Application program (응용 프로그램)

  - IDE (Integrated Development Environment)
    - Ex: Tornado – Development environment for VxWorks kernel
    - Linux – open source
    - 제품의 조기 출시

1.5 Processor Architecture and Organization

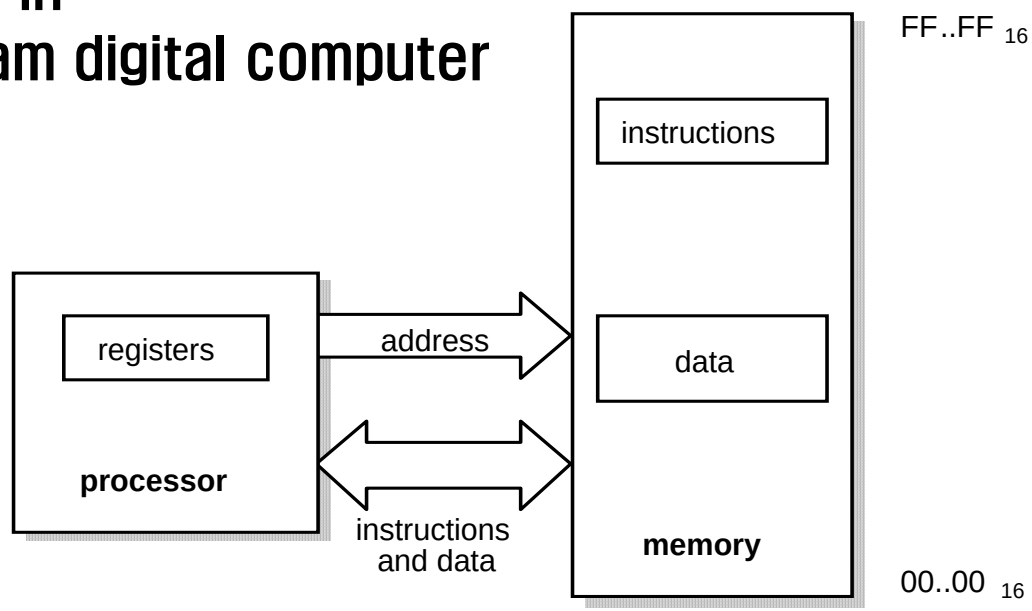
- All modern general-purpose computers
  - Principles of **stored-program digital computer**
  - Originated from Princeton Institute of Advanced studies (1940s)
  - Implemented in the 'Baby' machine at U Manchester (1948)
  - Technology advances
    - Vacuum tubes, transistors, IC, VLSI
  - New insights
    - Computer architecture
    - Computer organization

Processor Architecture and Organization (II)

- Computer architecture
  - Describes user's view of the computer
  - Instruction set, visible registers, memory management table structures, and exception handling model
- Computer organization
  - Describes the user-invisible implementation of the architecture
  - Pipeline structure, transparent cache, table-walking hardware, and translation look-aside buffer
- Advances
  - Virtual memory (1960s)
  - Transparent cache memory
  - Pipelining
  - RISC

Processor Architecture and Organization (III)

- Processor
  - Finite-state automaton that executes instructions held in a memory
  - The state of the system is defined
    - by the values held in the memory locations
    - The values held in certain registers within the processor
  - Fig 1.1 The state in a stored-program digital computer



Processor Architecture and Organization (IV)

- Stored-program computer
  - Keeps instructions and data in the same memory system
  - Allowing the instructions to be treated as data when necessary
  - Enables the processor itself to generate instructions which it can subsequently execute (self-modifying code)
    - Very difficult to debug
    - Not for programs in ROM
  - Loads a new program from disk (overwriting an old program)

- Computer applications
  - Programmability: universal
  - Embedded, desk top, mainframe

1.6 Abstraction in Hardware Design

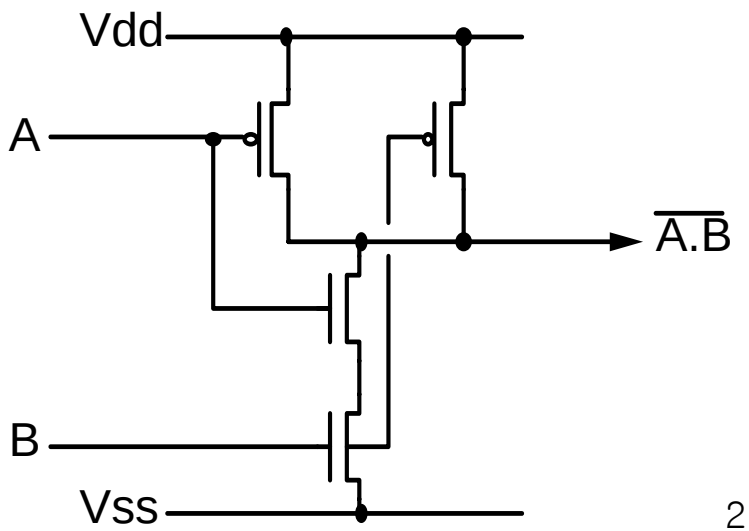
- Modern microprocessor
  - Several million transistors each of which can switch a hundred million times a second.
    - Ex) Scrolling a screen
  - A single error amongst these transitions is likely to cause the machine to collapse into a useless state

- Transistor
  - Electron movement described by quantum mechanics with probabilities
  - Voltage and current on its terminals perform amplification or switching

Abstraction in Hardware Design (II)

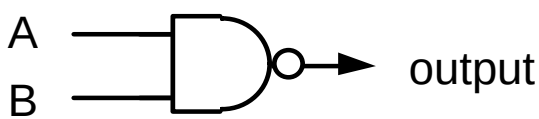
- Logic gates
  - A set of transistors is wired into a particular structure
  - CMOS (Complementary Metal Oxide Semiconductor) NAND gate
    - 2 inputs A, B and 1 output
    - If A and B are both near to Vdd (true), the output will be near to Vss.
    - If either A or B is near to Vss (false), the output will be near to Vdd.
    - NAND Boolean function:

$$Output = \neg(A \wedge B)$$



Abstraction in Hardware Design (III)

- Logic symbol
  - Symbol representing Boolean function in a circuit schematic
- Truth table
  - Describes the logic function of the gate
  - Encompasses everything that the logic designer need to know about the gate for most purposes
  - True='1'; False='0'
- Logic symbol and truth table of NAND gate:



Logic symbol

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

Abstraction in Hardware Design (IV)

- The gate abstraction
  - Greatly simplifies the process of designing circuits with great number of transistors
  - Removes the need to know that the gate is built from transistors
  - Implementation technology will affect the performance of the circuit, but it has no effect on its function
  - Supported by the transistor-level designer
    - As perfectly as possible
    - Isolate the logic circuit designer from the need to understand the transistor equations.

Abstraction in Hardware Design (V)

- Levels of abstraction
  - Essential behaviors and hide unnecessary details

- Typical hierarchy of abstraction at the hardware
  - Transistors
  - Logic gates, memory cells, special circuits
  - Single-bit adders, multiplexers, decoders, flip-flops
  - Word-wide adders, multiplexers, decoders, registers, buses
  - ALUs (Arithmetic-Logic Units), barrel shifters, register banks, memory blocks
  - Processor, cache and memory management organizations
  - Processors, peripheral cells, cache memories, memory management units
  - Integrated system chips
  - Printed circuit boards
  - Mobile telephones, PCs, engine controllers

Abstraction in Hardware Design (VI)

- Gate-level design
  - Libraries of useful functions
    - Adders, multiplexers, decoders: combinational circuits
    - Flip-flops and latches: sequential circuits
    - Registers
- Theoretic background
  - Boolean algebra
  - Combinational circuit design
  - Sequential circuit design