EE-334
Digital System Design

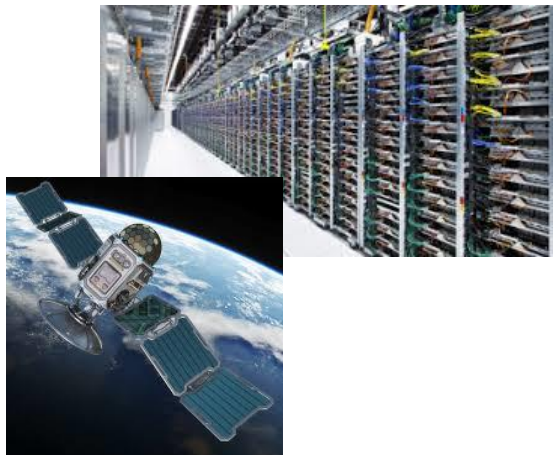
Digital Systems

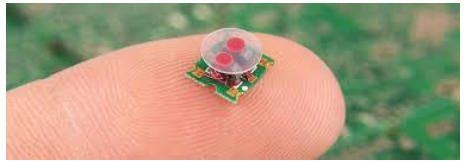
Andreas Burg

Digital Systems are Everywhere Today

- **Development** of digital systems is **complex and expensive**
- **Different markets** with different tradeoffs between volume and price
  - Different levels of complexity
  - Different volumes (from dozens to billions)
  - Different costs per device (cents to billions)

Economy of scale





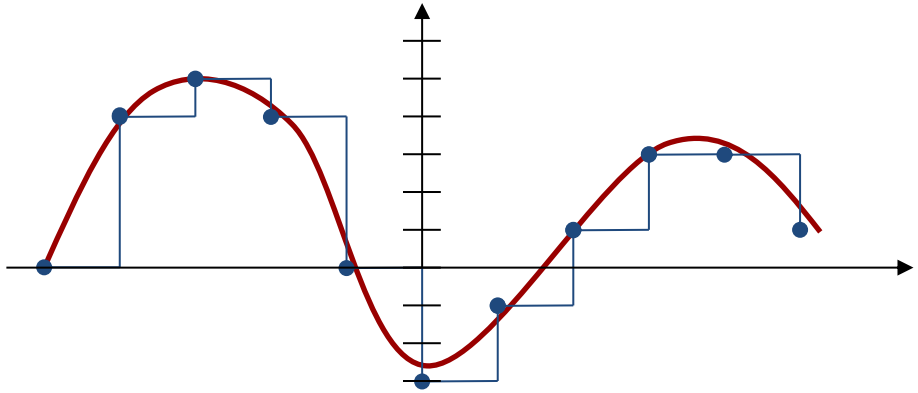
Low volume
high price

Medium volume
Medium price

High volume
Low price

What is a Digital System?

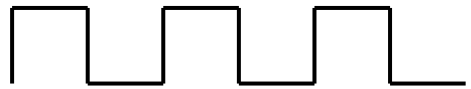
- As opposed to a analog systems, **digital systems** **represent everything with numbers** rather than physical quantities



- This automatically implies the need to
  - Discretize time
  - Discretize amplitude

- Discretization of amplitude: values expressed as integers
  - Numbers are typically represented in binary format
  - High immunity against noise

0100, 1001, 1110



- Discretization of time: relate every action to a (periodic) time reference, often called a clock

Why **Binary (Discrete Time)** Digital Systems?

- Operation of a digital system

Physical quantity (e.g., light)

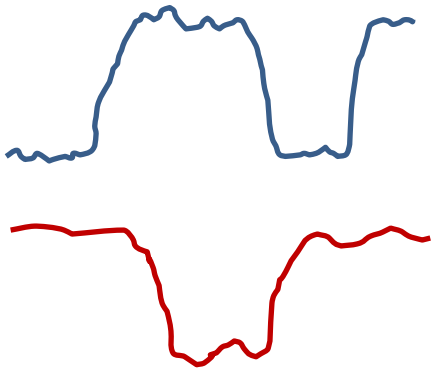
represented as numbers…

000234523000
000213233200
000723832400
002450482840
000323775900
000036893000

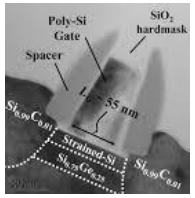
…in binary format with each digit …

00011
01000

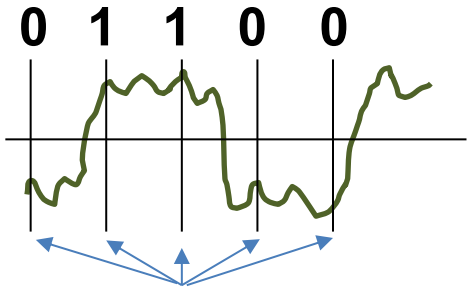
represented as physical quantity (e.g., voltage)



used in binary analog computations (DIGITAL LOGIC)



Leads to an analog result that is quantized back to a binary number



  - Computations are still performed with analog on physical quantities (voltages)
- **Binary representation** provides **robust against noise**
- **Discrete time** operation provides **robustness against uncertainties in time**

Ingredients for a Digital System

- Digital systems are often comprised of both hardware and software

Focus of this class

Physical components that operate on or store digital signals

Always required

Sequence(s) of instructions that direct the hardware to perform specific operations

Not always required

Key Differences Between Hardware and Software

Hardware

- **Operates on physical quantities** under the laws of physics
- Is a **composition of components** (abstracted devices)
  - Operates in parallel
- Fundamental components have **no inherent notion of time**
  - Ordered sequence of operations expressed by concatenating components
  - Time must be created artificially
- **Data shared though connections**

Software

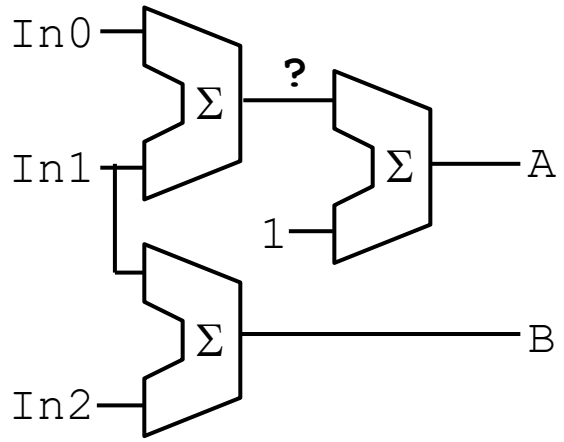
- **Operates on abstract variables** and numbers
- Is an inherently ordered **sequence of instructions**
  - Operates in sequentially
- Sequential nature of a **program implies order**
  - Ordered sequence of operations by executing them one after the other
  - Parallelism must be created artificially
- **Data shared through variables**

- Hardware and software design require **fundamentally different mindsets**

Key Differences Between Hardware and Software

Hardware

- Best **described as block diagram** (i.e., components / parallelism)



More operations naturally scale with resources

Software

- Best described as **sequence of instructions** (program)

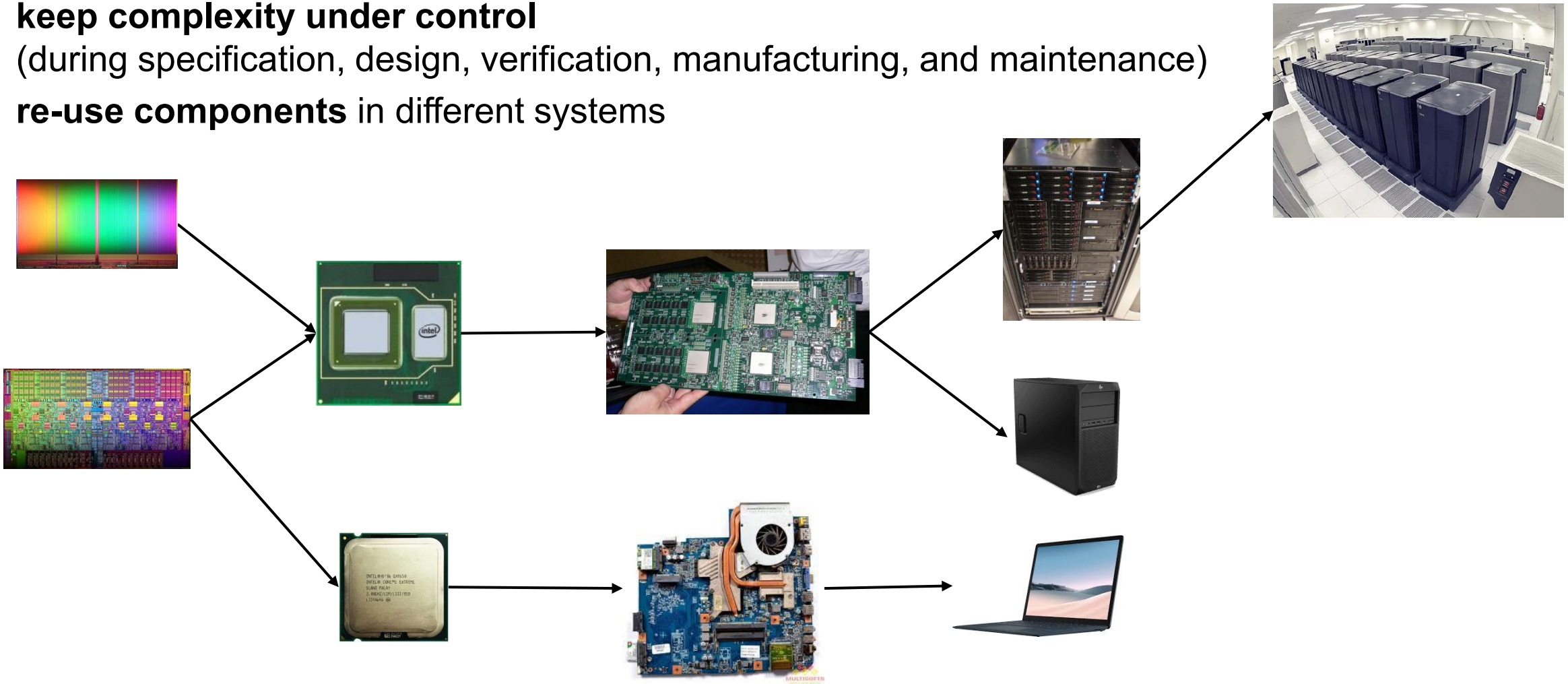
```
int In0=1;
int In1=2;
int In2=3;
int A, B;

A = In0 + In1;
A = A + 1;
B = In1 + In2;
```

More operations naturally scale required time

- Hardware and software design involves **fundamentally different tradeoffs**

Building Complex Digital Systems (Hardware)

- Digital Systems are built hierarchically to
  - **keep complexity under control**
    (during specification, design, verification, manufacturing, and maintenance)
  - **re-use components** in different systems

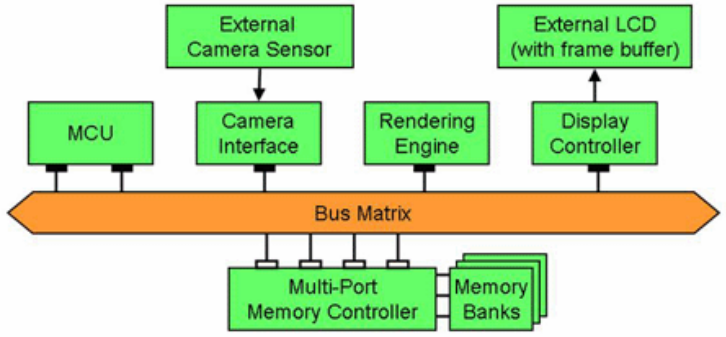


Designing with Abstraction

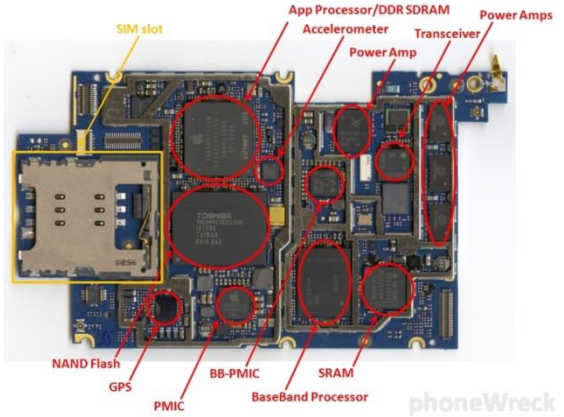
- Abstracting helps to **structure the design process**
- Different **levels of abstraction** are often **called VIEWs**

- **Three different views** are frequently used
  - **Functional view**: tasks (functions) that fulfill specific purposes
  - **Architectural view**: functional units (architectural components) needed to carry out the tasks
  - **Physical/technology view**: technologies to implement and connect architecture components

Functional view



Architectural view



Physical / technology view

Four Steps to Built a Digital System

- **Specification**: describe intended behavior and requirements (what not how)
- **Functional design**: define and partition behavior into tasks based on
  - Interaction between tasks
  - Communication requirements between tasks
  - Nature of the tasks (e.g., datapath or control)
- **Architectural design:** define the architectural building blocks of the system, how they are connected and which tasks they execute
- **Implementation:** choose implementation option for building blocks based on
  - Overall constraints (e.g., area, power, cost, …)
  - Requirements and nature of the task (e.g., computational load, memory needs, …)
  - Communication requirements between subtasks

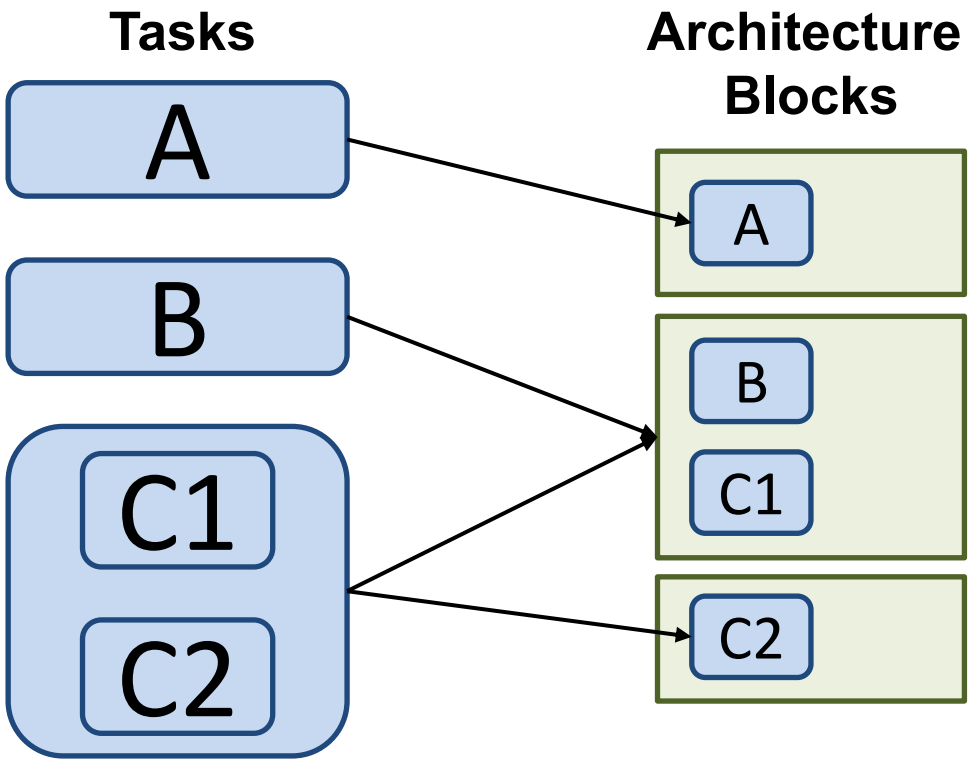
Architectural Building Blocks

- **Characteristics of a task decide** which **type of architecture** is required
- **Resources can be**
  - generic or specific for a task
  - custom built or already available

- **Examples** for resources
  - **I/O Interfaces**: .connect components or a system to the outside world
  - **Custom digital logic**: tailored to a very specific task
  - **Dedicated digital logic:** standard function for a specific, but common task
  - **Programmable processors**: execute software
  - **Storage arrays**: keep large amounts of data in a “hardware-efficient” way
    - Volatile memory (SRAM, DRAM)
    - Non-volatile memory
  - **Interconnect / busses**: provides communication architectural building blocks

Mapping Tasks to Architecture Building Blocks

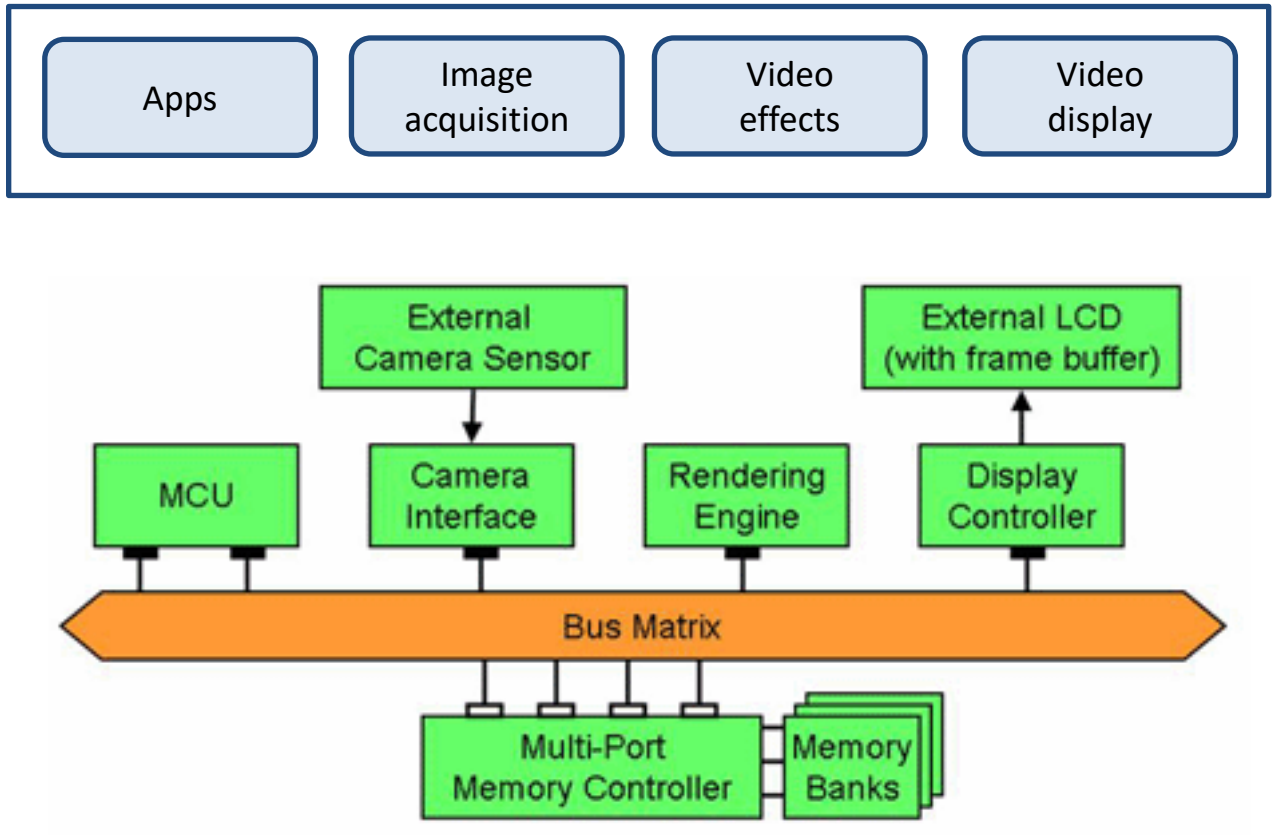
- **Assigning tasks** to architecture blocks is **NOT a 1:1 mapping**
  - A **single block** in an architecture can sometimes **handle or serve multiple tasks**
  - A **single task** (or different sub-tasks), carried out **on multiple (interacting) blocks**



- **Consideration**
  - Processing requirements and capabilities
  - Real-time requirements (latency)
  - Local memory requirements of a task
  - Capability of an architecture to carry out multiple tasks (parallel processing)
  - Flexibility of an architecture (ability to map tasks with different characteristics)
  - Communication requirements between tasks and sub-tasks
  - Communication interface protocols

Mapping Tasks to Architecture Building Blocks

- **Example:** An integrated camera system with live display and small apps

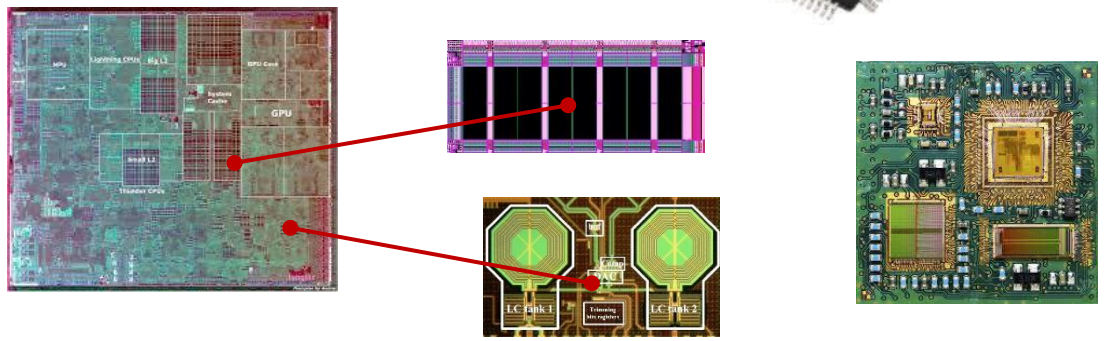


Implementation Options

- Often there are **many options for realizing** an **architectural building block**
- Implementation **options** often **differ significantly** in
  - Technical capabilities and performance
  - Ability to communicate with other components
  - Effort and cost during design and for the final product

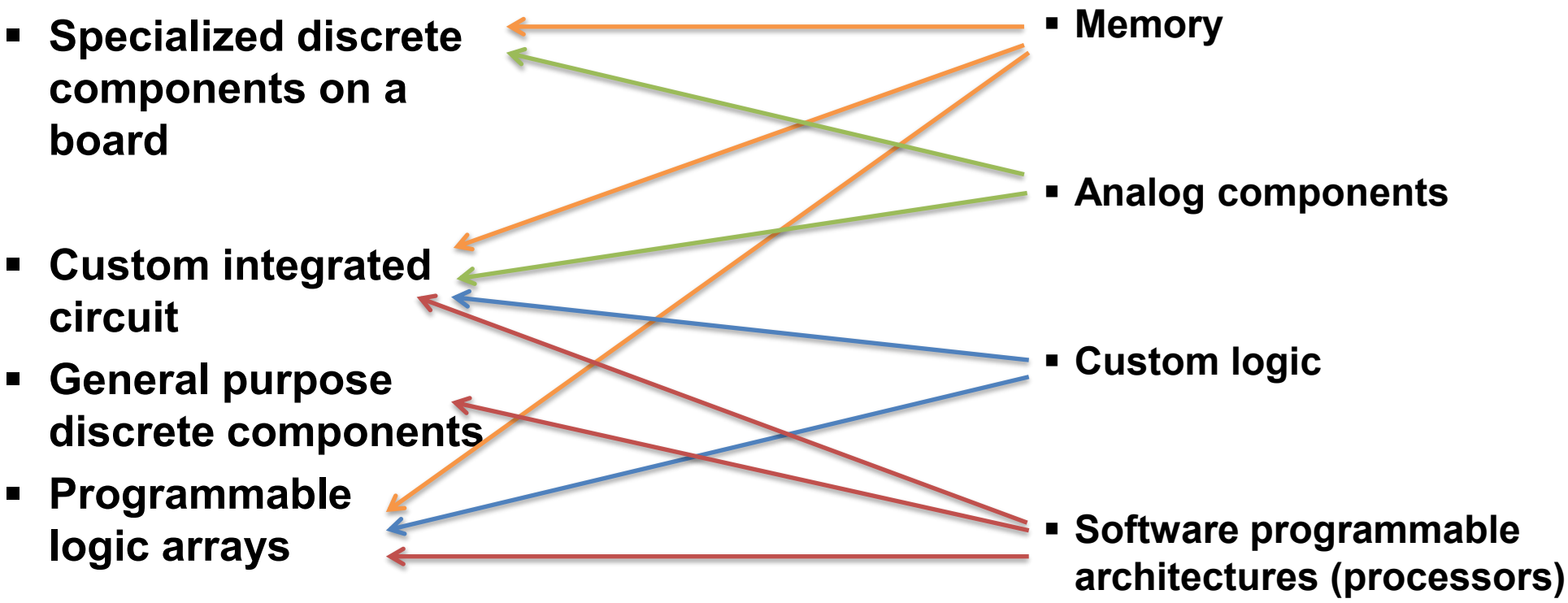
Often many options are not valid since they can not meet the requirements of the architecture

- **Examples** for resources
  - Discrete logic elements or components
  - Off-the-shelf complex components (ICs) or systems
  - Programmable logic devices (FPGAs)
  - Custom integrated circuits
    - Custom designed on-chip building blocks
    - Available IP components
    - Full custom analog blocks



Deciding on Implementation Options

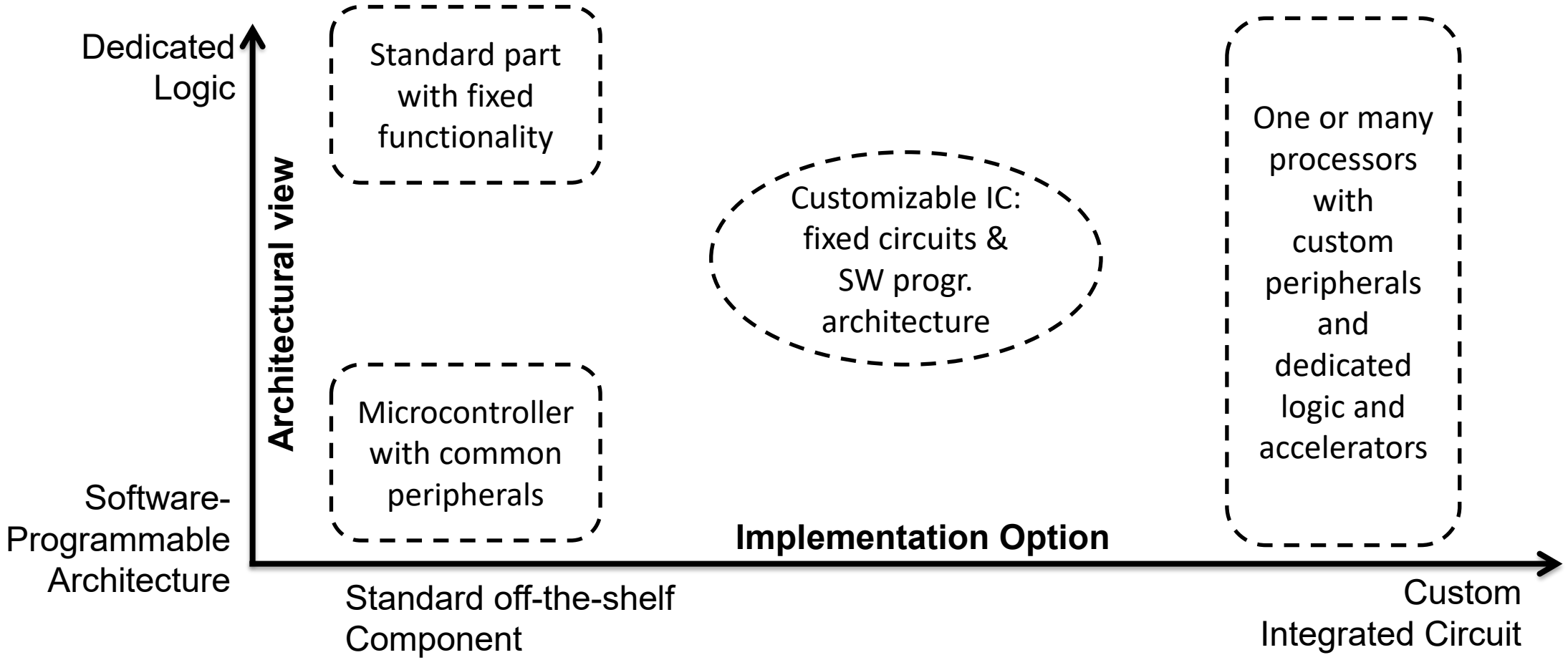
- Often **more than one option** exist **to implement** an architecture building block



- **Implementation options** often **impact** the **architecture** ➔ need **iterations**

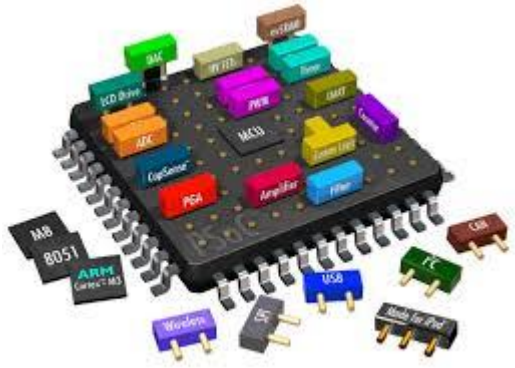
Examples for Different Implementation Options

- Different **implementation options** often **offer a large design space**



Trend Toward Higher Integration

- **Reducing** the **number of** physical **components**
  - **reduces** unit **cost**
  - **improves communication** between components
  - **reduces power** consumption
  - **improves reliability**

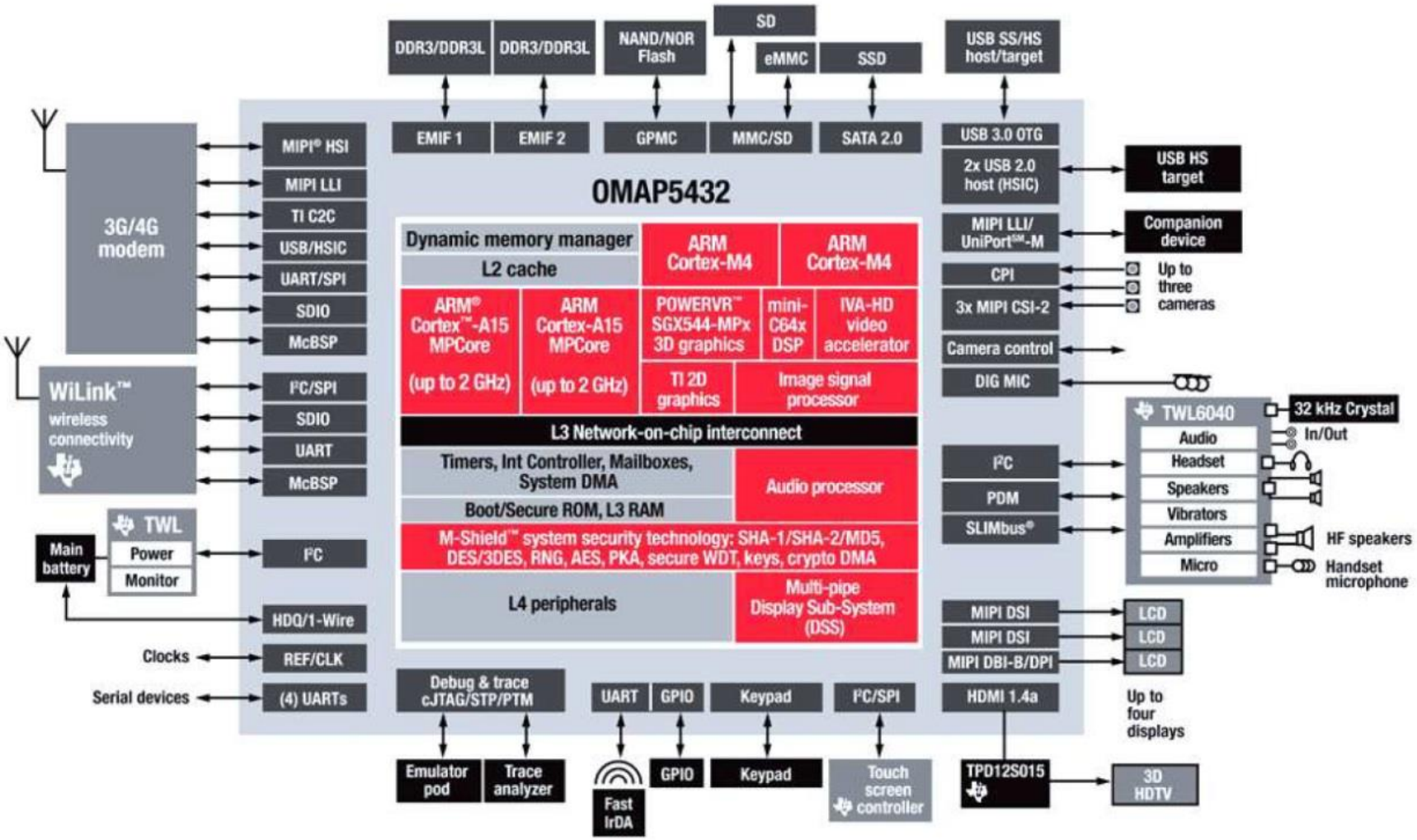


System on Chip (SoC)

- This **integration comes with** several **difficulties**
  - **Single technology can not always meet requirements of all** architectural **building blocks**
    - Non Volatile Memory (NVM) is not easily available in all CMOS technologies
    - Sensors or dense memories often require special technologies
    - FPGAs are not optimal for all typed of digital logic
    - …
  - **Dense integration often** requires custom design: expensive and time consuming

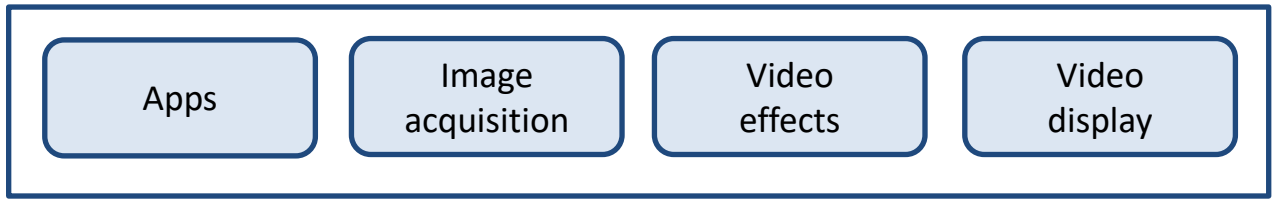
System Architecture Example: Mobile Phone

- **Architecture of 3rd generation phone** based on multiple integrated circuits with
  - TI System-on-Chip (SoC) as central component
  - External NVM
  - External DRAM
  - External wireless modem (RF/DSP)
  - External power management

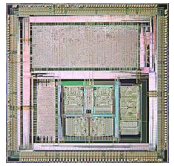


Mapping Architecture Blocks to Impl. Options

- **Example:** An integrated camera system with live display and small apps



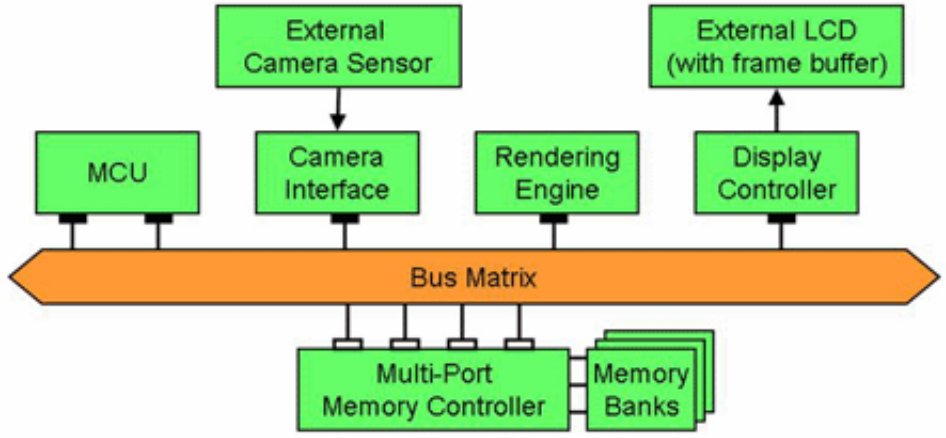
Camera module
I2C interface



Custom ASIC

Display adapter
I2C interface

uProcessor
SoC with standard
interfaces (I2C)



Programmable
Logic Device

External Memory

How to Select Available Components?

- Almost all **systems include** also some **standard components**

- Component selection and interaction with them requires understanding
  - **Functionality** and performance characteristics
  - **Interfaces**: communication protocols
  - Electrical specifications: different IO standards
  - Power supply and thermal requirements
  - Mechanical and thermal properties and requirements

- This **information** is found **in datasheets**
  - Datasheets for most components are available online
  - Datasheets often contain also other useful information
    - a rough block diagram of a circuit
    - application notes and examples


FTDI Chip

FT240X USB 8-BIT FIFO IC Datasheet
Version 1.2
Document No.: FT_000626 Clearance No.: FTDI# 259

Future Technology Devices International Ltd.
FT240X
(USB 8-BIT FIFO IC)

The FT240X is a USB to parallel FIFO interface with the following advanced features:

- Single chip USB to parallel FIFO bidirectional data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 2048 byte multi-time-programmable (MTP) memory, storing device descriptors and FIFO I/O configuration.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling glue-less interface to external MCU or FPGA.
- Data transfer rates up to 1Mbyte / second.
- 512 byte receive buffer and 512 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Configurable FIFO interface I/O pins.
- Synchronous and asynchronous bit bang interface options.
- USB Battery Charger Detection. Allows for USB peripheral devices to detect the presence of a higher power source to enable improved charging.
- Device supplied pre-programmed with unique USB serial number.
- USB Power Configurations; supports bus- powered, self-powered and bus-powered with power switching.
- Integrated +3.3V level converter for USB I/O.
- True 3.3V CMOS drive output and TTL input; operates down to 1V8 with external pull-ups.
- Configurable I/O pin output drive strength; 4 mA(min) and 16 mA(max).
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- +5V Single Supply Operation.
- Internal 3V3/1V8 LDO regulators
- Low operating and USB suspend current; 8mA (active-typ) and 125uA (suspend-typ).
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed capable.
- Extended operating temperature range; -40 to 85°C.
- Available in compact Pb-free 24 Pin SSOP and QFN-24 packages (both RoHS compliant).

Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Future Technology Devices International Ltd will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH United Kingdom. Scotland Registered Company Number: SC136640

Copyright © 2012 Future Technology Devices International Limited 1

Example: FTDI FTX240