# Leiden Embedded Research Center (LERC)

**Teddy Zhai**

Leiden Embedded Research Center
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands

# Outline

- LERC research team
- One of our main research problems
- Directions for Bachelor projects

# LERC group members

- Two senior staff members
  - Dr. Todor Stefanov (head)
  - Prof. Ed Deprettere

- 7 PhD students

# Embedded Systems-on-Chip (SoCs)

Embedded SoC = **Information processing system** that is

- ***embedded*** into a larger product
- ***application domain*** **specific** (*not* general purpose)
- ***stringent constraints***

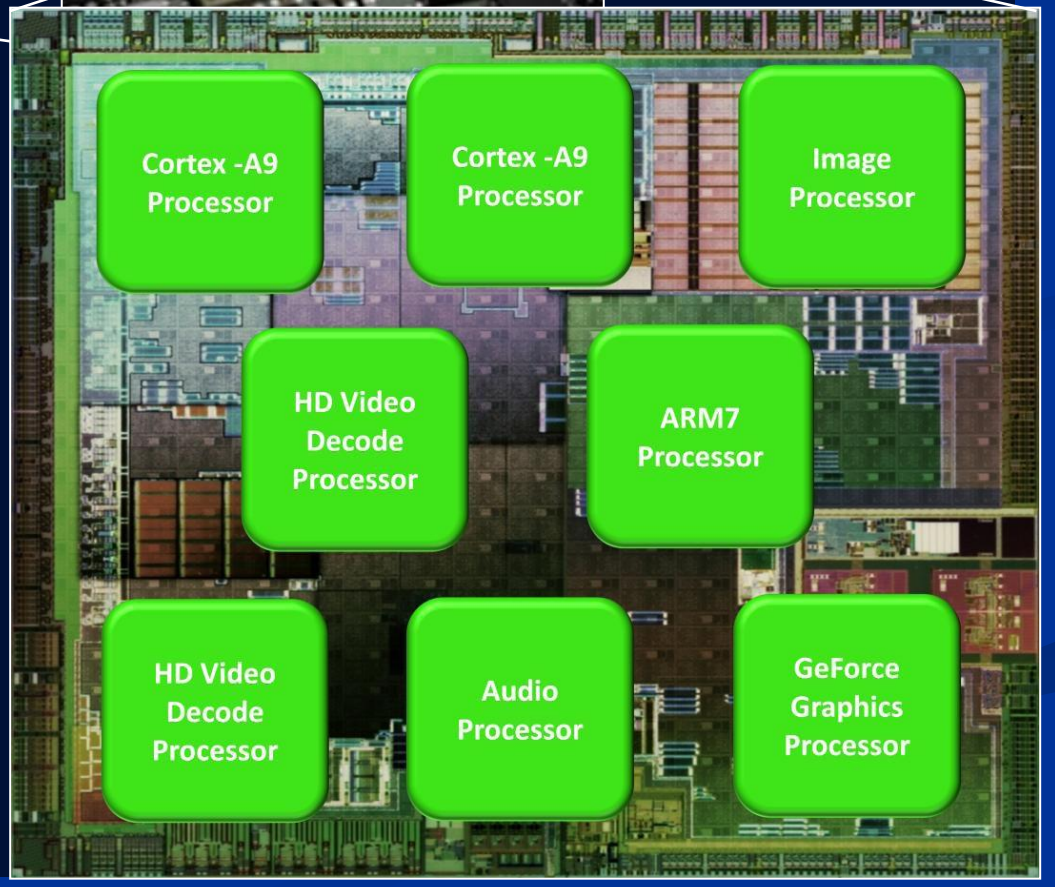
# Embedded SoCs are Everywhere!

Smart Phones


Tablets


Smart TVs

# Embedded SoCs are Everywhere!

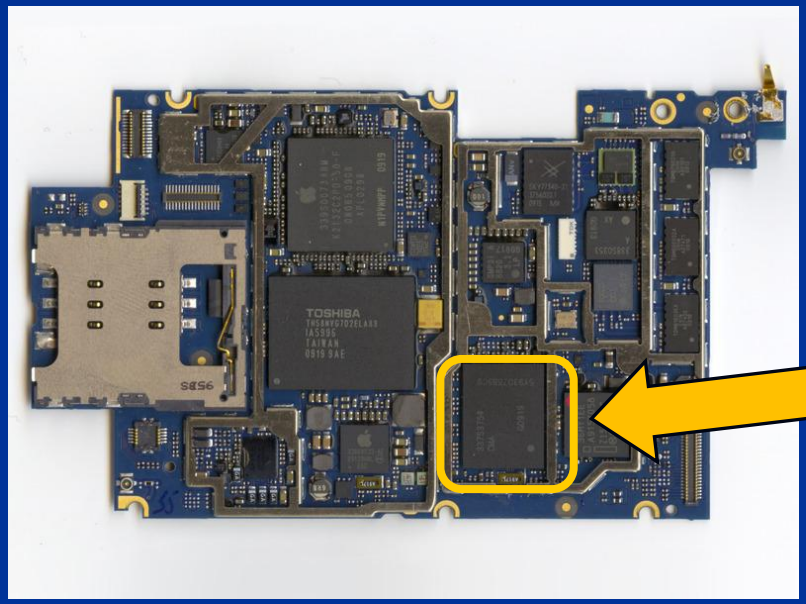




Multi-Processor
Embedded System-on-Chip

# What do we do in LIACS?

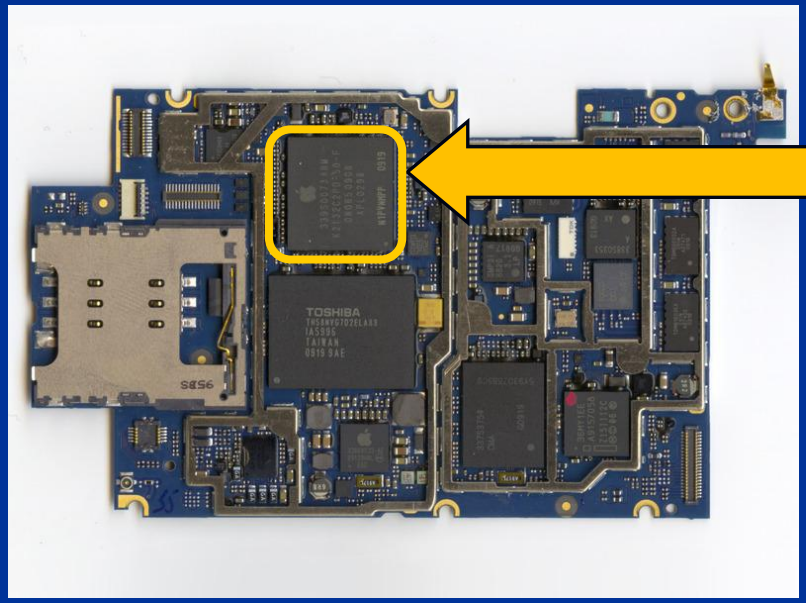
- Research on Embedded Systems-on-Chip (SoCs)
  - Focus on streaming applications
  - Programing and compilation techniques
  - Analysis of system properties (constraints)
  - Run-time support (OS)

# 4G wireless



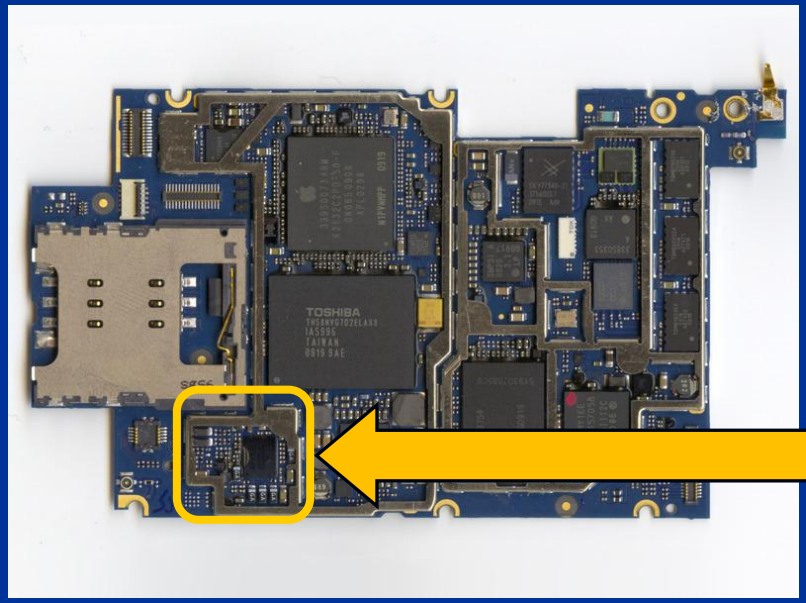

Anatomy of an iPhone

# Video coding




Anatomy of an iPhone

# Navigation




Anatomy of an iPhone

# Stringent constraints

- High performance
- Limited resource (CPU, memory)
- Hard real-time constraints
- Power/temperature constraints

# Our approach

## Application

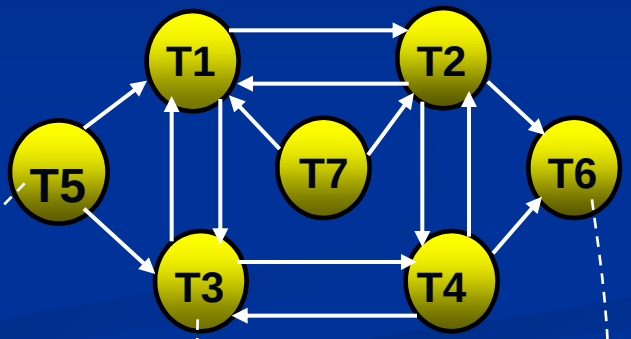
**EASY to specify**

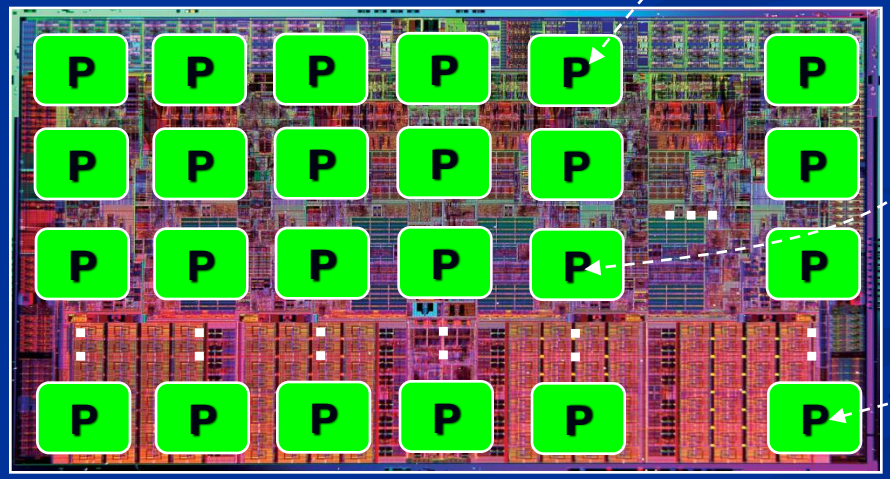
**Sequential Application Specification**

```
for j = 1:1:N,
  [x(j)] = Comp1( );
end
for i = 1:1:K,
  [y(i)] = Comp2( );
end
for j = 1:1:N,
  for i = 1:1:K,
[y(i), x(j)] = Comp3(y(i),x(j));
  end
end
for i = 1:1:K,
  [Out(i)] = Comp4( y( i ) );
end
```

Parallelization

**Parallel Application Model**




**EASY to map**

Mapping

1000-Processor Embedded System-on-Chip

# Directions for Bachelor projects

- Application
- Compiler
- Design tools
- Real-time operating system

# Application modeling

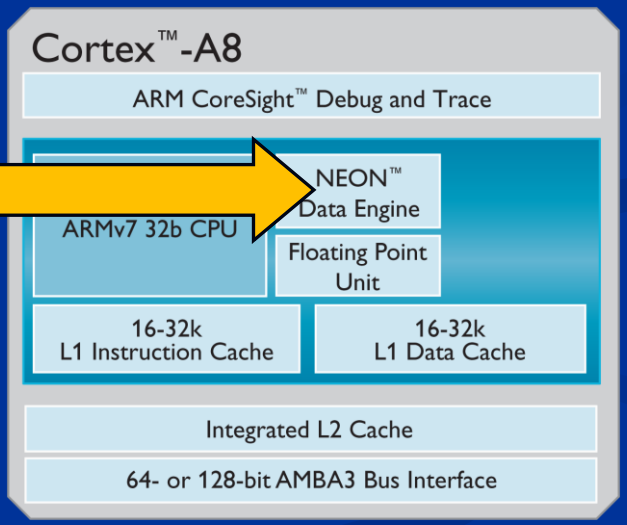
- Formal parallel models for
  - 4G LTE
  - High Efficiency Video Coding (HEVC)

- To facilitate
  - Compilation (parallelization)
  - Analysis

# Compiler

- Automatic parallelization
  - Loop transformation
  - Vectorization

```
for j = 1:1:N,
  [x(j)] = Comp1( );
end
for i = 1:1:K,
  [y(i)] = Comp2( );
end
for j = 1:1:N,
  for i = 1:1:K,
[y(i), x(j)] = Comp3(y(i),x(j));
  end
end
for i = 1:1:K,
  [Out(i)] = Comp4( y( i ) );
end
```




- Automatic code generation

# Analysis/Optimization

- Analysis of system properties: analytical or simulation-based
  - Throughput/latency
  - Power/temperature
- Single or multi-objective optimizations to
  - Maximize performance
  - Minimize power consumption/temperature
  - Maximize resource utilization

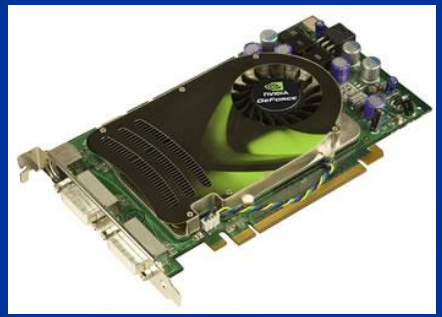
# Real-time Operating Systems

- Evaluate different RTOSs and scheduling algorithms
- Develop new scheduling algorithms
  - Power/temperature-efficient
  - Resource-efficient

# Tool/equipment's support

- Access to research and/or commercial tools and SW/HW platforms.

- Example of HW platforms
  - Xilinx FPGAs
  - NVIDIA GPUs
  - …



# Basic requirements

- Knowledge in C
  - Other high-level languages are useful
- Interested in system architecture
  - Compilation toolchain
  - OS
  - Hardware architecture (CPU, memory, interconnect)
- Comfortable with low-level details

# Thank you

# Detailed Bachelor projects

- Modeling, analysis, and optimizations of Embedded SoCs
  - Modeling using variety of Models of Computations
    - Process Networks, Dataflow graphs, etc.
  - Analytical or Simulation-based Analysis and Verification to check if SoC requirements are met:
    - Functional: consistency, deadlock-free, etc.
    - Non-functional: performance, power consumptions, cost, etc.
  - Single or Multi-objective Optimizations to:
    - Maximize SoC performance
    - Minimize SoC power consumption
    - Minimize/Maximize SoC resource utilization
    - …

# Detailed Bachelor projects

- Program Code Analysis and Transformations
  - Automated parallelization of program code into tasks
  - Transformations  of program code to increase/decrease parallelism (i.e. number of tasks)
- Mapping of program code onto Embedded Multi-processor SoCs
  - Efficient generation of task code for processors in C/C++
  - Efficient allocation of tasks code on processors
  - Efficient scheduling of multiple tasks on a processor
  - Converting task code into fast HW circuits
  - …