Who we are

- Prof. Dr. Akash Kumar
  - Head at Chair for Processor Design (from 2015)
  - Assistant Professor at National University of Singapore, Singapore
  - PhD from Eindhoven University of Technology, Netherland
  - India

- Dr. Tuan D. A. Nguyen
  - Postdoc at Chair for Processor Design (from 2018)
  - PhD from National University of Singapore, Singapore
  - Some industrial experiences in SoC for datacenter and high-frequency trading domain
  - Vietnam
Chair for Processor Design

- [https://cfaed.tu-dresden.de/pd-about](https://cfaed.tu-dresden.de/pd-about)
- Helmholtzstrasse 18, 3rd Floor room BARIII73-74, BARIII76
- Research interests:
  - Analysis, design and resource management of novel predictable low-power (multi-)processor architectures
  - Approximate computing
  - Thermal-aware 3D architectures
  - Reliable/fault-tolerant multiprocessor systems
Agenda

1. Why and What is “Embedded Hardware”? 
2. Module plans 
3. Integrated Circuit 
4. Circuit Taxonomy 
5. Circuit Design Implementation Styles 
6. Circuit Models 
7. Synthesis
1. Why and What is “Embedded Hardware”? 
2. Module plans 
3. Integrated Circuit 
4. Circuit Taxonomy 
5. Circuit Design Implementation Styles 
6. Circuit Models 
7. Synthesis
Why this “hardware design”? 

- You may wonder:
  - “I’m a software guy, why do I need to take this module?”
  - “What is hardware?”
Heterogeneous Cloud Computing

- Scale-out Horizontally
- Scale-out Vertically

Request Service A

OpenStack / Hypervisor

Service A

VM0

VM1 VM2 VM3 VMn

Scale-out Horizontally

© Akash Kumar
Cloud with FPGAs

- Amazon F1
- Microsoft Catapult
- ARM
- Deephi
- NGCodec
- Napatech
- Maxeler
- Falcon Computing
- Baidu
- Huawei
- Plunify
- IBM
- Accelize
- Analog Design
- Ericsson

© Akash Kumar
## Cloud with FPGAs

Table 1. Comparison of CPU-only vs. Brainwave-accelerated TP1 and DeepScan DNN models in Bing production.

<table>
<thead>
<tr>
<th></th>
<th>Bing TP1</th>
<th>Bing DeepScan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU-only</td>
<td>Brainwave-accelerated</td>
</tr>
<tr>
<td>Model details</td>
<td>GRU 128x200 (x2) + W2Vec</td>
<td>LSTM 500x200 (x8) + W2Vec</td>
</tr>
<tr>
<td></td>
<td>9 ms</td>
<td>0.850 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End-to-end latency</td>
<td>9 ms</td>
<td>0.850 ms</td>
</tr>
<tr>
<td>per Batch 1 request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at 95%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Objective of this Module
Already “high” about cloud?
Why embedded systems?

- In fact, FPGAs on the cloud are some sort of “embedded systems”
Embedded Systems Definition

- An embedded system is nearly any computing system (other than a general-purpose computer) with the following characteristics:
  - Single-functioned
    - Typically, is designed to perform predefined functions
  - Tightly constrained
    - Tuned for low cost
    - Single-to-fewer components based
    - Performs functions fast enough
    - Consumes minimum power
  - Reactive and real-time
    - Must continually monitor the desired environment and react to changes
  - Hardware and software co-existence
Embedded Systems Examples
Embedded Systems Example

DragonFly+: An FPGA-based quad-camera visual SLAM system for autonomous vehicles (Shaoshan Liu - PerceptIn)

- Single-functioned
  - Typically, is designed to perform predefined function

- Tightly constrained
  - Tuned for low cost
  - Single-to-fewer components based
  - Performs functions fast enough
  - Consumes minimum power

- Reactive and real-time
  - Must continually monitor the desired environment and react to changes

- Hardware and software co-existence
Major Factors to be Considered

- Time-to-market
- Technology
- Performance
- Cost
- Power consumption
- Reliability
- Testability
- Availability of CAD tools, libraries, IP's
- ...
Major Factors to be Considered

- Time-to-market
- Technology
- Performance
- Cost
- Power consumption
- Reliability
- Testability
- Availability of CAD tools, libraries, IP's
- ...

Major Factors to be Considered

- Time-to-market
- Technology
- Performance
- Cost
- Power consumption
- Reliability
- Testability
- Availability of CAD tools, libraries, IP's
- ...


Major Factors to be Considered

- Time-to-market
- Technology
- Performance
- Cost
- Power consumption
- Reliability
- Testability
- Availability of CAD tools, libraries, IP’s
- ...
Major Factors to be Considered

- Time-to-market
- Technology
- Performance
- Cost
- Power consumption
- Reliability
- **Testability**
- Availability of CAD tools, libraries, IP's
- ...

https://www.evaluationengineering.com/instrumentation/article/13013319/systems-and-software-address-testability
Major Factors to be Considered

- Time-to-market
- Technology
- Performance
- Cost
- Power consumption
- Reliability
- Testability
- Availability of CAD tools, libraries, IP's
- ...

https://www.semiwiki.com/forum/content/3534-non-separation-power-performance.html
We will examine the design methodologies to implement computations (algorithms) on platforms.

tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;
Simplified and General Embedded System Design Methodology

- Algorithm Functional Modeling
  - Problem Partitioning
  - SW Func. Model
  - SW/HW Interface
  - Hw Func. Model
  - Architectural Synthesis
  - Logic/Physical Synthesis
- Sw. Dev
- Application Source Code
- Processor
- SW/HW Interface
- Application Specific Hardware
Who Contribute to Embedded System Designs

Algorithm Functional Modeling

Problem Partitioning

Sw Func. Model

SW/HW Interface

Hw Func. Model

Sw. Dev

Application Source Code

Processor

Backend (UX/UI)

Backend (Database, Network, etc.)

Backend (Driver, OS, HW/SW Interface, etc.)

QA/QC

Frontend (UX/UI)

Backend (Database, Network, etc.)

Backend (Driver, OS, HW/SW Interface, etc.)

QA/QC

Frontend (Macro/Micro Architecture)

Backend (Synthesis, Timing Analysis, Placement and Routing)

Hardware Design (Board)

Verification/Validation

Processors

Application Specific Hardware

Who Contribute to Embedded System Designs

1. Frontend (UX/UI)
2. Backend (Database, Network, etc.)
3. Backend (Driver, OS, HW/SW Interface, etc.)
4. QA/QC

Algorithm Functional Modeling

Problem Partitioning

Sw Func. Model

SW/HW Interface

Hw Func. Model

Sw. Dev

Application Source Code

Processor

Application Specific Hardware

Frontend (Macro/Micro Architecture)

Backend (Synthesis, Timing Analysis, Placement and Routing)

Hardware Design (Board)

Verification/Validation

Who Contribute to Embedded System Designs

1. Frontend (UX/UI)
2. Backend (Database, Network, etc.)
3. Backend (Driver, OS, HW/SW Interface, etc.)
4. QA/QC
Agenda

1. Why and What is “Embedded Hardware”?
2. Module plans
3. Integrated Circuit
4. Circuit Taxonomy
5. Circuit Design Implementation Styles
6. Circuit Models
7. Synthesis
Textbook For This Module

- Giovanni De Micheli,
- *Synthesis and Optimization of Digital Circuits*,
- Please focus on Chapters 1, 3, 4, 7, 9, and 11
- Classic textbook on VLSI Design Automation.
Lecture Plan

- **Part 1 (6 lectures):**
  - General introduction to embedded hardware system design
  - FPGA architecture and EDA
  - System Specifications and Modeling
  - System Specifications with High-level Language
  - Architectural Behavioral Synthesis

- **Part 2 (6 lectures):**
  - Logic Synthesis
  - Logic Minimization
  - Technology Mapping
  - Physical Synthesis

- **Schedule:** Thursdays, 11.10-12.40 (Computer Science Faculty, room APB E009)

- **Dates:** 4.4. / 11.4. / 18.4. / 25.4. / 2.5. / 9.5. / 16.5. / 23.5. / 6.6. / 27.6. / 4.7. / 11.7.
Lecture Plan

- Next semester, **practical module** will be offered as follow-up to get the hands-on on the FPGA
Self-study
Groups of 1-2 students
Select/propose topics
Meet every 2 weeks
Foundation for the practical module next semester (learn one, get three 😊)

Plan
- First month: topic definition, techniques exploration, literature survey, ideas
- Second month: implementation
- Third month: report, demonstration

Register here or later via email tuan_duy_anh.nguyen1@tu-dresden.de
- Header: [ESD19][Seminar]....
1. Why and What is “Embedded Hardware”?
2. Module plans
3. Integrated Circuit
4. Circuit Taxonomy
5. Circuit Design Implementation Styles
6. Circuit Models
7. Synthesis
Integrated Circuit

- A piece of silicon
- Transistors

CMOS Cross Section

Schematic diagram of 2-input NAND gate using CMOS

2-input NAND gate symbol
Integrated Circuit

Here is a look at the inside of a 1965 transistor radio made with 7 transistors.

In 2007 one large memory IC contained a sufficient number of transistors to make more than one billion of these radios.

http://bugbookmuseum.blogspot.com/2015/07/discrete-component-electronics-to.html
Moore's Law

Moore's law is the observation that the number of transistors in a dense integrated circuit doubles about every two years.

(Gordon Moore, the co-founder of Fairchild Semiconductor and CEO of Intel)
CAD – Computer-Aided Design Tools

- Or EDA – Electronic Design Automation tools
- Computer programs that automate parts of the design process

https://en.wikipedia.org/wiki/Physical_design_(electronics)
CAD – Computer-Aided Design

- Intel 4004 (1971)
  - 10 um
  - 2300 Transistors

- Intel Dual-core GT2 Skylake (2015)
  - 14 nm
  - 1.7 Billions Transistors

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client)#Dual-core
Agenda

1. Why and What is “Embedded Hardware”?
2. Module plans
3. Integrated Circuit
4. **Circuit Taxonomy**
5. Circuit Design Implementation Styles
6. Circuit Models
7. Synthesis
Circuit Taxonomy - Material

- CMOS
- BiCMOS
Circuit Taxonomy - Behavior

- Analog
  - Continuous

- Digital
  - Discrete

ADC – Analog Digital Converter

DAC – Digital Analog Converter
ADC - DAC
Digital Circuit

- Synchronous
  - Operations are synchronized to one (or more) global signal - clock

- Asynchronous
  - No clock is needed
Circuit Taxonomy - Summary

IC

Material

Digital

Analog

Asynchronous

Synchronous

Combinational

Sequential

?
1. Why and What is “Embedded Hardware”?
2. Module plans
3. Integrated Circuit
4. Circuit Taxonomy
5. Circuit Design Implementation Styles
6. Circuit Models
7. Synthesis
Circuit Design Implementation Styles

- Full-custom vs. Semi-custom design style
- Affected by conflicting factors
  - Projected volume
  - Expected pricing
  - Circuit performance
  - Reliability
  - Time
  - Etc.
Full-Custom Design

- All (or most) of the design processes are handcrafted
  - Specifying the layout of each individual transistor and the interconnections between them
- Tremendous efforts and time
- However, very high-quality circuits
- Only benefit with extremely high volume
- Now only found in crucial CPU blocks which are reused across a large number of product lines
Semi-custom Design

- The circuit primitives are > transistors
- Blocks or gates
- Relies on EDA tools to assemble them
Semi-custom Design Variants

- **Cell-based**
  - Standard Cells
  - Macro Cells

- **Array-based**
  - Masked-programmable
  - Field-programmable

- Memory
- Arithmetic
- Etc.
- Gate arrays
- Sea of gates
- Anti-fused based
- Memory-based
Possible Platforms

- General Purpose: CPUs, GPUs
- Reconfigurable: FPGAs
- Application Specific: ASICs

Agenda

1. Why and What is “Embedded Hardware”? 
2. Module plans 
3. Integrated Circuit 
4. Circuit Taxonomy 
5. Circuit Design Implementation Styles 
6. Circuit Models 
7. Synthesis
A model of a circuit is an abstraction
- A representation with features of interest without associated details
- Synthesis is a process of generating a more detailed model from a less detailed one
Levels of Abstractions

Architectural Level
- Algorithm Functional Modeling
- Problem Partitioning
- Sw Func. Model
- SW/HW Interface
- Hw Func. Model
- Architectural Synthesis

Logic Level
- Logic/Physical Synthesis

Geometrical Level
- Application Specific Hardware
Example

tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;

Architectural Level  Logic Level  Geometrical Level
Views of a Model

Behavioral View

```
tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;
```

Structural View

Physical View
Views and Levels of Abstractions

- **Views**
  - Behavioral
  - Structural
  - Physical

- **Abstractions**
  - Architectural level
  - Logic level
  - Geometrical level

Gajski and Kahn's Y-chart
Views and Levels of Abstractions

tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;

Architectural Level

Logic Level

Geometrical Level
Views and Levels of Abstractions

Architectural Level
Logic Level
Geometrical Level

Structural View

Behavioral View

Physical View

Architectural Level
Logic Level
Geometrical Level

MEM
MULT
CONTROL
ADD

© Akash Kumar
Views and Levels of Abstractions

Architectural Level
Logic Level
Geometrical Level

Behavioral View
Structural View
Physical View

Architectural Level
Logic Level
Geometrical Level
1. Why and What is “Embedded Hardware”?
2. Module plans
3. Integrated Circuit
4. Circuit Taxonomy
5. Circuit Design Implementation Styles
6. Circuit Models
7. Synthesis
Synthesis

- Synthesis can be seen as a set of transformation between two axial views
  - Architectural synthesis
  - Logic synthesis
  - Physical synthesis

Gajski and Kahn’s Y-chart
Architectural Synthesis

- Identify hardware resources that can implement the operations
- Schedule the execution of the operations
- Bind operations to resources
  - Data Path and Control Unit
    - Data Path: interconnection of resources
    - Control Unit: issuing control signals to the data path according to the schedule
tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;
tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;

One ALU as either an adder or a multiplier is available
One adder and one multiplier are available

schedule 1  schedule 2
tmp0 = a + b;
tmp1 = c * d;
tmp2 = tmp0 * tmp1;
tmp3 = tmp0 + tmp2;
e = tmp2 * tmp3;

One ALU as either an adder or a multiplier is available

One adder and one multiplier are available
Logic Synthesis

- Transforming the high-level description of the logic to the actual combination of logic gates and their connections through a series of logic optimizations.
Logic Synthesis - Input

- Logic-level model of a circuit can be provided by a state transition diagram or finite-state machine (FSM), circuit schematic or equivalent by High-level Description Language (HDL)
- Specified by designers or synthesized from an architectural level model
Logic Synthesis - Input

One ALU as either an adder or a multiplier is available

Schedule 1

Schedule 2

State Machine

HDL

```hdl
ap_ready <= ap_const_logic_1;
ap_return <=
    tmp_fu_28_p2 when (out_write_assign_fu_40_p0(0) = '1')
else
    tmp_1_fu_34_p2;
out_write_assign_fu_40_p0 <= (0=>op, others =>'1');
tmp_1_fu_34_p0 <= in2;
tmp_1_fu_34_p1 <= in1;
tmp_1_fu_34_p2 <= std_logic_vector(signed(tmp_1_fu_34_p0) + signed(tmp_1_fu_34_p1));
tmp_fu_28_p0 <= in2;
tmp_fu_28_p1 <= in1;
tmp_fu_28_p2 <= std_logic_vector(IEEE.numeric_std.resize(unsigned(
    std_logic_vector(signed(tmp_fu_28_p0) * signed(tmp_fu_28_p1))), 32));
```
Logic Optimization

```
ap_ready <= ap_const_logic_1;
ap_return <=
    tmp_fu_28_p2 when
        out_write_assign_fu_40_p0(0) = '1'
    else
tmp_l_fu_34_p2;
out_write_assign_fu_40_p0 <= (0 xor op, others =>"-");
tmp_l_fu_34_p0 <= in2;
tmp_l_fu_34_p1 <= in1;
tmp_l_fu_34_p2 <= std_logic_vector(signed(
tmp_l_fu_34_p0) + signed(tmp_l_fu_34_p1));
tmp_fu_28_p0 <= in2;
tmp_fu_28_p1 <= in1;
tmp_fu_28_p2 <= std_logic_vector(IEEE.
numeric_std.resize(unsigned(
std_logic_vector(signed(tmp_fu_28_p0) *
signed(tmp_fu_28_p1)), 32));
```

Logic synthesis will do various optimizations to simplify the logic expressions
Logic Synthesis - Example
Physical Synthesis

- Technology mapping
- Generating the layout of the chip
- Place the gates/transistors
- Route (connect) the components
Technology Mapping

- A process of taking a logic expression and converting that representation into a new logic representation that can be implemented by cells in standard cell library or FPGA resources

- Determining a collection of various covers which are decided from pattern recognition

- Choosing a particular set such that a desired metric is optimized such as area or delay.
Technology Mapping - Example
Layout and Placement

- Placement is to map the shape of a cell to a physical place in the layout.
- Cells that have many connections with each other should be placed close to each other so as to minimize delays.
Playout and Placement - Example
Routing

- Routing is to map the logic connection between cells to a physical interconnect in the layout.
- The physical route for each logic connection should be as short as possible to minimize delays.
Routing - Example

Control Logic

ALU
Summary

- There are many contributing factors in designing embedded hardware systems
- HW and SW operations are optimized to work hand-in-hand
- Hardware platforms can be general purpose, reconfigurable or application specific depending on the requirements
- There are many steps involved during the design process
- EDA tools play a very important role