Research at LAP

Processor Architecture Lab
Current Research

1. Novel FPGAs
2. FPGAs for Software Programmers
3. Elastic Circuits
4. CNN Vision Accelerator
5. Circuit Sketching & Debugging
6. SAT for Logic Synthesis
7. Automatic DPA Hardening

VLSI Design and Automation
Programming Paradigms and Methodologies
Reconfigurable Computing

Computer Architecture and Engineering
Rethinking FPGA Architectures

**Idea:** Replace Look-Up Tables with And-Inverter Cones (AICs) as basic logic blocks

**Goal:** Increase device efficiency by renouncing some excess of flexibility

---

**Look-Up Tables**
- Full flexibility
- Slow
- Large

**And-Inverter Cones**
- Limited flexibility
- Fast
- Small

---

Parandeh-Afshar et al. *Rethinking FPGAs: Elude the flexibility excess of LUTs with And-Inverter Cones*. FPGA 2012  **Best Paper Award**
1 Exploring FPGA Architectures

Problem: To study FPGA architectures, you need good retargetable FPGA toolsets (synthesis, mapping, P&R), such as VTR, but you also need good models of the transistor-level primitives (LUTs, crossbars, novel logic elements) and of their combinations.

Idea: Leverage a common standard-cell VLSI flow to pick and optimize transistor-level primitives and to produce accurate models for the complete architectures.

Difficulties: (1) Standard cells are not designed the same ways FPGAs circuitry is and (2) VLSI tools are not built for this purpose.

Yet, FPRESSO shows how this can be achieved with useful fidelity.

fpresso.epfl.ch
Efficient Hardware Design from Domain-Specific Languages

**Idea:** Build domain-specific architectures and optimizations into DSLs and into frameworks to create DSLs (e.g., Delite)


George, Lee, Novo, Rompf, Brown, Sujeeth, Odersky, Olukotun, and Ienne. *Hardware system synthesis from domain-specific languages*. FPL 2014

Generating Multi-Module Designs from Computational Patterns

**Idea:** Decompose high-level applications (e.g., written using a DSL) into computational patterns to automatically build multi-module designs achieving high-performance.

**Problems:**
1. Synchronization
2. Load Balancing
3. System Composition

**Performance Benefit of Multi-Module Design**

FPGA in the Cloud

**Idea:** FPGA as a cloud resource, simultaneously shared among multiple users (like CPUs)

**Problem:** Lack of protection and virtualization facilities to share FPGA resources

- Hardware virtualization support
- Dynamic instantiation of accelerators via partial reconfiguration
- Access to hardware occurs via system calls to runtime manager (similar to OS)

Sharing resources across multiple concurrent applications reduces the overall execution time compared to sequential exclusive use
Problem: Most CGRAs are statically scheduled like VLIWs. (1) It is hard to develop efficient schedules and (2) such schedules are very sensitive to latency variability.

Idea: An array of elastic components interconnected by an FPGA-like routing network → the superscalar of the CGRAs!

Elastic circuits mimic the dynamic processing of asynchronous circuits in a synchronous context and provide control-flow synchronization primitives (EB, EF, MG, JO, BR...)

A complete tool chain to optimize, map, place, and route C programs on the Elastic CGRA

J. Huang, Y. Huang, Chen, Ienne, Temam, and Wu. A low-cost memory interface for high-throughput accelerators. CASES 2014
Y. Huang, Ienne, Temam, Chen, and Wu. Elastic CGRAs. FPGA 2013
Dynamically Scheduled HLS

**Problem:** Current HLS tools produce **statically scheduled** circuits
- Control and data dependencies degrade pipeline performance
- Worst-case schedule when dependency analysis cannot provide conclusive information
- In general, same limitations as VLIW processors (good for regular DSP applications, bad otherwise)

**Idea:** Create **dynamically scheduled** circuits where operations are executed as the operands are ready
- Data dependencies are resolved on-the-fly
- Opens the door to speculative execution (achieving what is common in superscalar processors)

**Example: Designing a simple processor using HLS**

```c
while (done == 0) {
    curr_inst = inst_mem[pc];
    opcode = curr_inst & 0b111111;
    operand_1 = reg_file[(curr_inst >> 6) & 0b11111];
    operand_2 = reg_file[(curr_inst >> 11) & 0b11111];
    immediate = (curr_inst >> 6) & 0xffff;
    switch (opcode) {
    case 0x04: // ADD opcode
        result = operand_1 + operand_2;
        reg_file[(curr_inst >> 6) & 0b11111] = result;
        pc++;
        break;
    case 0x0c: // AND opcode
        result = operand_1 & operand_2;
        reg_file[(curr_inst >> 6) & 0b11111] = result;
        pc++;
        break;
    case 0x0d: // JR (relative jump) opcode
        pc += immediate;
        break;
    case 0x34: // HALT opcode
        done = 1;
        break;
    }
}
```

**HLS with static scheduling**

**HLS with speculation and dynamic scheduling**

Josipović, Brisk, and Ienne. From C to Elastic Circuits. Asilomar SSC 2017
### 4 Accelerating CNNs for Vision and Classification

**Problem:** Convolutional and Deep Neural Networks (CNNs and DNNs) are extremely effective for some classification tasks but are terribly computationally demanding and poorly suited for GPGPUs.

**Idea:** Design a programmable accelerator suitable for integration in a standard embedded camera chipset (no DRAM, limited SRAM, etc.).

**Result:** About 30x more performance than a typical GPU for less than 1/4500th of the energy.

**New Challenges:** Blood cell analysis requires classification of images at speeds that are 100-1000x better than the best server GPGPUs and at a small fraction of the energy.

**Real-Time Cell Classifier for Early Diagnosis**

(cooperation with IMEC)
Improving Circuit Design with a “Fill-in-the-Blank” EDA Tool

**Problem:** Humans are good at big-picture design, but not the details

**Idea:** Let software build the circuit from a designer’s sketch

---

```scala
var io = {
    val i = Bits(IN, w)
    val o = Bits(OUT, w)
}

val div = MY_CONSTANT;
io.o := io.i / div;
```

**Designer specifies naïve functionality and provides incomplete optimized RTL design**

**Software tool solves a problem to find how to fill in holes left by the designer in the optimized design**

**Designer gets back completed, optimized, guaranteed-correct circuit in RTL**

---

Becker, Novo, and Ienne. *SKETCHILOG: Sketching combinational circuits*. DATE 2014

Becker, Novo, and Ienne. *Automated circuit elaboration from incomplete architectural descriptions*. Asilomar SSC 2013
Automated Circuit Debugging

**Problem:** Debugging is expensive, even though the same bug types appear often.

**Idea:** Instrument circuits with common fixes. Solver finds if some combination actually works.

---


---

### Library

- \( l_1: \ e := x + y \)
  - \( l_{1,1}: \ e := x + y \)
  - \( l_{1,2}: \ e := x - y \)
- \( l_2: \ e := x \& y \)
  - \( l_{2,1}: \ e := x \& y \)
  - \( l_{2,2}: \ e := x \mid y \)

### RTL Source

```vhdl
else if (op == OP_SH)
  o=shift(b, a[5:0]);
```

### Problem Formulation

Software tools determine suspicious RTL, apply matching error rules, and find fixing combination(s)

### Interpretation

- **rtl/alu.v@29.11**
  - Signal ‘b’ should be ‘a’
- **rtl/alu.v@29.13**
  - Signal ‘a’ should be ‘b’

**Designer gets back meaningful error diagnosis exactly describing the problem and necessary fix.**

---

Buggy circuit design and library of common RTL errors provided to software suite
6 SAT Methods for Logic Synthesis

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Fast Generation of LEXSAT Assignments

Problem: Generate a satisfiable assignment with the smallest (or greatest) integer value for a given variable order (a LEXSAT assignment)

Idea: Use the concept of the binary search algorithm to determine the LEXSAT assignment

Results: 2.4x faster generation of a single assignment, 6.3x faster generation of multiple assignments

Application 1: SAT-based computation of canonical Sum of Products (SOPs)

LEXSAT enables generating minterms in a deterministic order

Application 2: Heuristic NPN classification for large functions (with up to 194 variables)

LEXSAT enables comparing the integer value of two truth tables


Soeken, Mishchenko, Petkovska, Sterin, Ienne, Brayton, and De Micheli. Heuristic NPN Classification for Large Functions Using AIGs and LEXSAT. SAT 2016 Best Paper Award Nominee

Petkovska, Mishchenko, Soeken, De Micheli, Brayton, and Ienne. Fast Generation of Lexicographic Satisfiable Assignments: Enabling Canonicity in SAT-based Applications. IWLS 2016 Best Student Paper Award Nominee

Petkovska, Mishchenko, Novo, Owaida, and Ienne. Progressive Generation of Canonical Sums of Products Using a SAT Solver. IWLS 2016
Constrained Interpolation for Guided Logic Synthesis

Problem: Craig interpolation reconstructs a target function $f$ using random base functions from the given set $G$ and often omits some wanted base functions (up to 60%).

Carving Interpolation is a novel method to impose a base function $g_i$:

$$ f = h(g_1, \ldots, g_n) = g_i \cdot I_{g_i} + \overline{g_i} \cdot I_{\overline{g_i}} $$

$f$ is reconstructed using a Shannon expansion of two constrained interpolants built for $g_i = 1$ and $g_i = 0$, respectively.

Useful for ECO and some logic synthesis algorithms.

Omits only 0.15% of the wanted base functions.
Automated Side-Channel Vulnerability Discovery and Hardening

**Goal:** Detect the sensitive parts of a given hardware/software implementation against side-channel attacks and protect these parts using proper countermeasures.

**Step I:** Detection of sensitive operations
- **Input Software Implementation:**
  - `sbci r26,0xfd`
  - `ld r25,X`
  - `movw r18,r26`
  - `subi r18,0x4f`

**Step II:** Identification of transformation targets
- **Sensitive Parts:**
  - `sbci r26,0xfd`
  - `ld r25,X`
  - `movw r18,r26`
  - `subi r18,0x4f`

- **Targets for Protection:**
  - `sbci r26,0xfd`
  - `ld r25,X`
  - `movw r18,r26`
  - `subi r18,0x4f`

**Step III:** Code transformation
- **Protected Implementation:**
  - `sbci r26,0xfd`
  - `lds r23,rnd`
  - `mov r25,r23`
  - `ld r25,X`
  - `lds r23,rnd`
  - `mov r18,r23`
  - `mov r19,r23`
  - `movw r18,r26`
  - `subi r18,0x4f`

Recent Topics

**Logic Synthesis for Arithmetic**

EDAA Outstanding Dissertation Award 2010 and Patrick Denantes Memorial Award 2011
Verma, Brisk, and Ienne. *Iterative Layering: Optimizing arithmetic circuits by structuring the information flow*. ICCAD 2009
Verma and Ienne. Improving XOR-dominated arithmetic circuits by exploiting dependencies between operands. ASPDAC 2007. **Best Paper Award Nominee**

**Improving FPGAs for Datapaths**

Huang, Ienne, Temam, Chen, and Wu. Elastic CGRAs. FPGA 2013

**Extending Flash Lifetime**

Jimenez, Novo, and Ienne. *Phoenix: Reviving MLC blocks as SLC to extend NAND flash devices lifetime*. DATE 2013
Less Recent Topics

**VLSI Design and Automation**

- **Self-Calibrating Design**
  - *EDAA Outstanding Dissertation Award 2006*
  - Worm, Thiran, and Ienne. *Designing robust checkers in the presence of massive timing errors*. IOLTS 2006
  - Worm, Ienne, Thiran, and De Micheli. *On-Chip self-calibrating communication techniques robust to electrical parameter variations*. IEEE D&T 2004

- **Customizable Processors**
  - Atasu, Pozzi, and Ienne. *Automatic Application-Specific Instruction-Set Extensions under Microarchitectural Constraints*. DAC 2003. **Best Paper Award**

- **Portable Reconfigurable Accelerators**

**Programming Paradigms and Methodologies**

- **Computer Architecture and Engineering**