Research at LAP

Paolo Ienne
Processor Architecture Lab

EPFL
Current Research

1. Novel FPGAs
2. Dynamically Scheduled High-Level Synthesis
3. Advanced Memory Systems
4. Accelerators
5. Circuit Sketching & Debugging
6. SAT for Logic Synthesis
7. Automatic DPA Hardening

- VLSI Design and Automation
- Programming Paradigms and Methodologies

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.
**Problem:** FPGA interconnect is highly programmable, to be able to support a large number of user designs; this negatively impacts its performance. **Can we improve the interconnect?**

**Idea:** Introduce *fast and cheap direct connections*

**Challenge:** Where to insert them so that many user designs can benefit from them and others are not excessively damaged?

---


---

**Problem:** To **effectively use direct connections**, new CAD algorithms are needed

---

Michal Servit Best Paper Award

---


---

**Problem:** FPGA interconnect is highly programmable, to be able to support a large number of user designs; this negatively impacts its performance. **Can we improve the interconnect?**

**Idea:** Introduce *fast and cheap direct connections*

**Challenge:** Where to insert them so that many user designs can benefit from them and others are not excessively damaged?

---


---

**Problem:** To **effectively use direct connections**, new CAD algorithms are needed

---

Michal Servit Best Paper Award

---

Problem: Interconnect delays scale poorly (R and C grow quickly and wires become “slower”)

Idea: Quantify the impact of technology scaling on FPGA performance and derive ways to mitigate it

Challenge: Devise precise enough models to capture the impact, but simple enough for architectural exploration

Potential switch types

Problem: Complex influence of technology scaling on FPGA performance calls for effective automation of exploration of aspects of FPGA architecture insufficiently explored before

Idea: Leverage the algorithm of the FPGA router to let it decide what needs to be fabricated
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
- Control and data dependencies are resolved on-the-fly

for (i=0; i<N; i++) {
    hist[x[i]] = hist[x[i]] + weight[i];
}

Josipović, Brisk, and Ienne. *From C to Elastic Circuits*. Asilomar SSC 2017
Josipović, Sheikhha, Guerrieri, Ienne, and Cortadella. *Buffer Placement and Sizing for High-Performance Dataflow Circuits*. FPGA 2020 **Best Paper Award**
2 An Out-of-Order Memory Interface for Spatial Computing

**Problem:** Conventional Load-Store Queue allocation policy is not suitable for spatial computing
- In a processor, decoding conveys the correct sequential order of requests at the memory interface
- Spatial circuits do not have instructions or a program order

**Idea:** Allocate groups of memory accesses depending on the control flow of the application
- Groups: sequences of accesses which are statically predefined
- If one access of a group executes, all other accesses belonging to the same group will eventually execute

Groups correspond to basic blocks in HLS: all accesses in a basic block are going to take place if the control flow enters the basic block

---

Speculation in Dataflow Circuits

**Challenge:** Execute certain operations before it is known that they are correct or required
- Increase parallelism in loops where the condition takes long to compute
- Improve performance of circuits limited by potential memory dependencies

```c
float d = 0.0; x = 100.0; int i = 0;
while (d < x) do {
    d = a[i] + b[i];
    c[i] = d;
    i++;
}
```

0: a[0]=50.0; b[0]=30.0
1: a[1]=40.0; b[1]=40.0
2: a[2]=50.0; b[2]=60.0 → exit

**Idea:** Contain **speculation in a region** of the circuit delimited by special components
- Issue speculative tokens (pieces of data which might or might not be correct)
- Squash and replay in case of misspeculation

**Nonspeculative schedule:** conservatively wait for loop condition
- 1: d = a[0] + b[0]; c[0] = d; d0 = x?
- 2: d2 = a[1] + b[1]; c[1] = d2; d1 = x?

**Speculative schedule:** tentatively start another loop iteration
- 1: d0 = x?
- 2: d1 = a[0] + b[0]; c[0] = d1; d1 = x?
- 3: d2 = a[1] + b[1]; c[1] = d2; d2 = x?

Josipović, Guerrieri, and Ienne. *Speculative Dataflow Circuits*. FPGA 2019
An Elastic Coarse Grain Reconfigurable Array

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

Miss-Optimized Memory Systems (MOMSes)

Problem: Effective DRAM bandwidth is more than one order of magnitude smaller than peak bandwidth when accesses from accelerators are irregular and narrow (e.g., sparse linear algebra, graph traversal).

Idea: Instead of trying to increase the hit rate (with costly caches), focus on better handling misses. That is, build nonblocking caches with 1000’s of Miss Status Holding Registers (MSHRs) instead of the usual 10-20.

Challenge: How to implement efficiently the associative structure needed for 1000’s of MSHRs?!
Large-Scale Graph Processing on FPGAs with MOMS

Problem: Show that MOMSes can be useful for practical FPGA accelerators in the cloud

Intuition: No need for software defined data movement, dynamic data movement (“caches”) can be made to work if designed appropriately

<table>
<thead>
<tr>
<th>Platform</th>
<th>Ext. mem. bandwidth</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work, FabGraph</td>
<td>FPGA</td>
<td>64 GB/s</td>
</tr>
<tr>
<td>Gunrock</td>
<td>GPU</td>
<td>900 GB/s</td>
</tr>
<tr>
<td>Ligra, GraphMat</td>
<td>CPU</td>
<td>233 GB/s</td>
</tr>
</tbody>
</table>

Results:
- PageRank:
  - 3x faster than state-of-the-art on FPGA
- PageRank, Strongly Connected Components:
  - competitive with state-of-the-art on CPU and GPU
  - 1.1x to 15.3x more bandwidth and power efficiency
- Single Source Shortest Path
  - competitive with SotA on CPU
  - GPU is much faster but can only handle the smallest benchmarks!

Asiatici and Ienne. *Large-Scale Graph Processing on FPGAs with Caches for Thousands of Simultaneous Misses.* ISCA 2021
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 $\frac{1}{4500}$th 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

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 combinatorial 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.


---

**Problem Formulation**

Library

\[ l_1: e := x + y \]

\( (l_{1,1}): e := x + y \)

\( (l_{1,2}): e := x - y \)

\[ l_2: e := x \& y \]

\( (l_{2,1}): e := x \& y \)

\( (l_{2,2}): e := x \mid y \)

---

Buggy circuit design and library of common RTL errors provided to software suite

 RTL Source

\[ \begin{align*}
\text{else if(op == OP_SH)} \\
& o=\text{shift}(b,a[5:0]);
\end{align*} \]

---

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

---

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

---

**rtl/alu.v@29.11**

Signal ‘b’ should be ‘a’

**rtl/alu.v@29.13**

Signal ‘a’ should be ‘b’
## 6 SAT Methods for Logic Synthesis

### 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

<table>
<thead>
<tr>
<th></th>
<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>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **smallest SAT assignment**
- **greatest SAT 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, Novo, Owaida, and Ienne. *Progressive Generation of Canonical Sums of Products Using a SAT Solver*. IWLS 2016 **Best Student Paper Award Nominee**
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.

Petkovska, Novo, Mishchenko, and Ienne. Constrained interpolation for guided logic synthesis. ICCAD 2014 Best Paper Award Nominee
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

```
sbci r26,0xfd
ld  r25,X  *
movw r18,r26 *
subi r18,0x4f
```

**Input Software Implementation**

**Step II:** Identification of transformation targets

**Independent targets**

```
sbci r26,0xfd
ld  r25,X
movw r18,r26
subi r18,0x4f
```

**Sensitive Parts**

```
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
```

**Sensitive Parts**

**Dependent targets**

```
sbci r26,0xfd
lds  r23,rnd
mov  r25,r23
ld  r25,X
```

**Targets for Protection**

**Step III:** Code transformation

**Local transformation**

**Global transformation**

```
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
```

**Protected Implementation**

Bayrak, Regazzoni, Novo, Brisk, Standaert, and Ienne. *Automatic application of power analysis countermeasures.* IEEE TCOMP, February 2015

Bayrak, Regazzoni, Novo, and Ienne. *Sleuth: Automated verification of software power analysis countermeasures.* CHES 2013
**Selected Not So Recent Topics**

**VLSI Design and Automation**

- **Improving FPGAs for Datapaths**

**Logic Synthesis for Arithmetic**

  - 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**

**Computer Architecture and Engineering**

**Programming Paradigms and Methodologies**

- **Extending Flash Lifetime**
Selected Even 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


Dubach, Vuletić, Pozzi, and Ienne. Enabling unrestricted automated synthesis of portable hardware accelerators for virtual machines. CODES 2005

Vuletić, Pozzi, and Ienne. Seamless hardware-software integration in reconfigurable computing systems. IEEE D&T 2005

Vuletić, Pozzi, and Ienne. Dynamic prefetching in the virtual memory window of portable reconfigurable coprocessors. FPL 2004

Programming Paradigms and Methodologies

Computer Architecture and Engineering