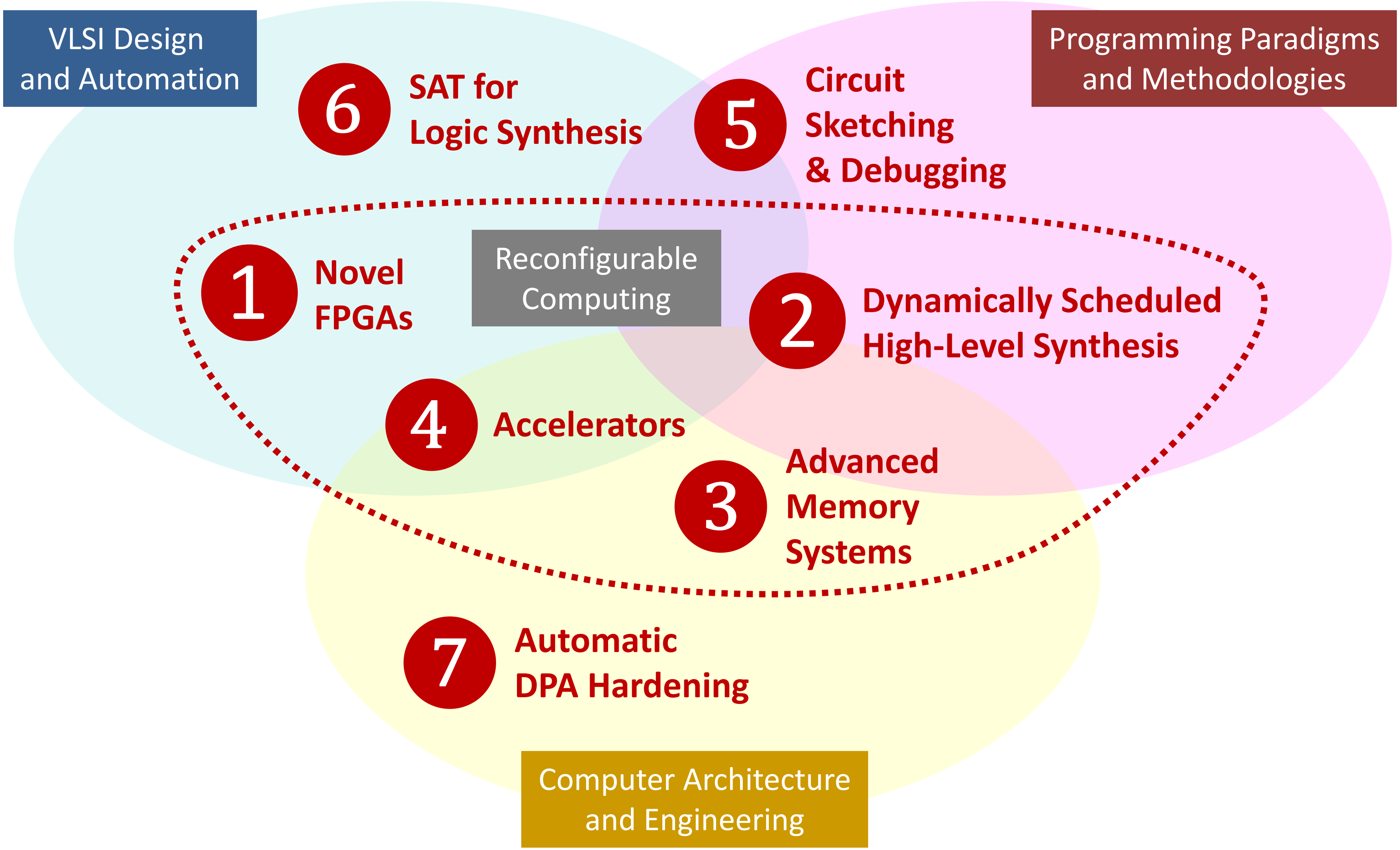
Research at LAP

Paolo Ienne

Processor Architecture Lab

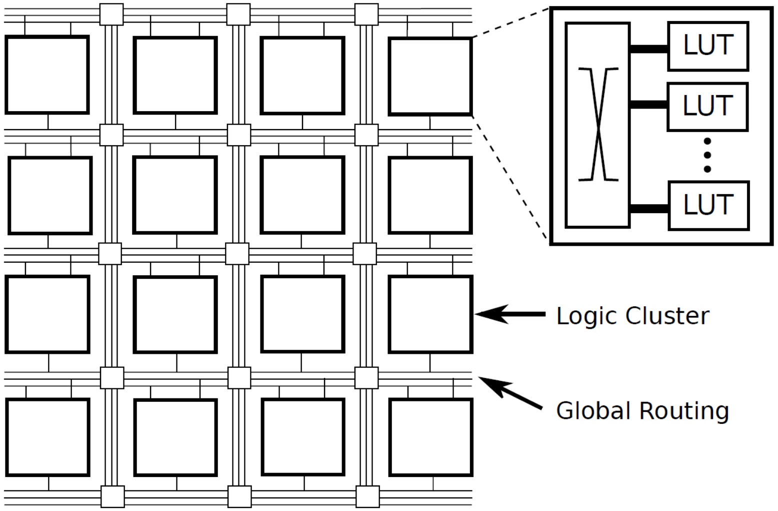
EPFL

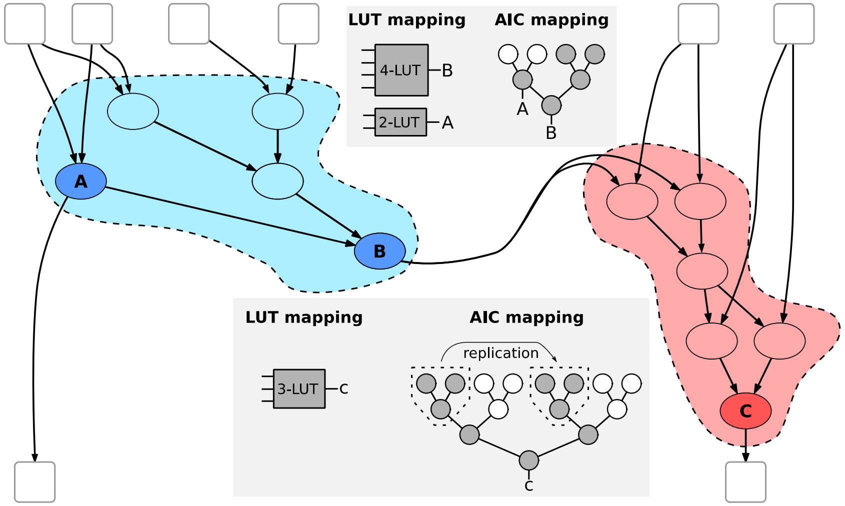
Current Research



VLSI Design and Automation
Programming Paradigms and Methodologies
Computer Architecture and Engineering
Reconfigurable Computing
6 SAT for Logic Synthesis
5 Circuit Sketching & Debugging
1 Novel FPGAs
2 Dynamically Scheduled High-Level Synthesis
4 Accelerators
3 Advanced Memory Systems
7 Automatic DPA Hardening

1 Rethinking FPGA Architectures

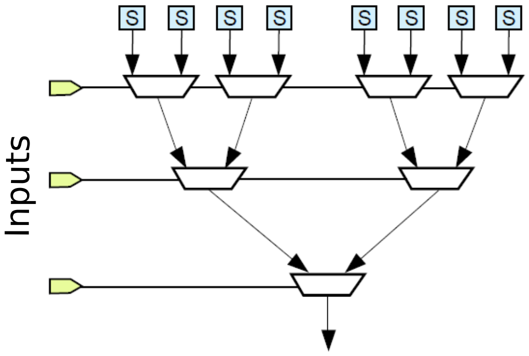


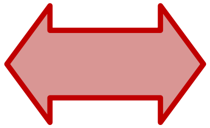


Idea: Replace Look-Up Tables with **And-Inverter Cones (AICs)** as basic logic blocks
Goal: Increase device efficiency by renouncing some excess of flexibility

Look-Up Tables

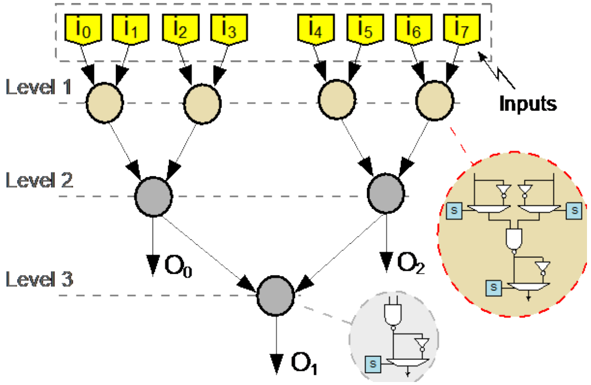
Full flexibility
Slow
Large





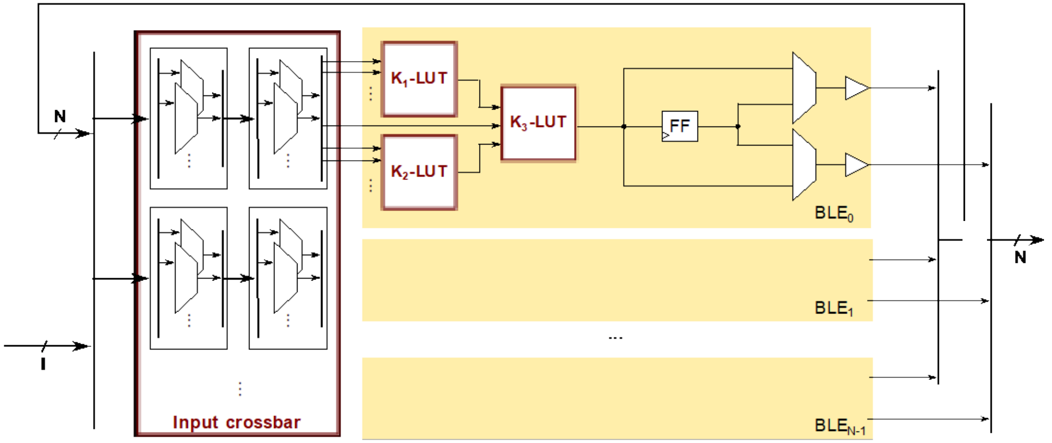
And-Inverter Cones

Limited flexibility
Fast
Small



Jiang, Zgheib, Lin, Novo, Huang, L. Yang, H. Yang, and Ienne. *A Technology Mapper for Depth-Constrained FPGA Logic Cells*. FPL 2015
Zgheib, Yang, Huang, Novo, Parandeh-Afshar, Yang, and Ienne. *Revisiting And-Inverter Cones*. FPGA 2014
Parandeh-Afshar et al. Rethinking FPGAs: *Elude the flexibility excess of LUTs with And-Inverter Cones*. FPGA 2012 **Best Paper Award**

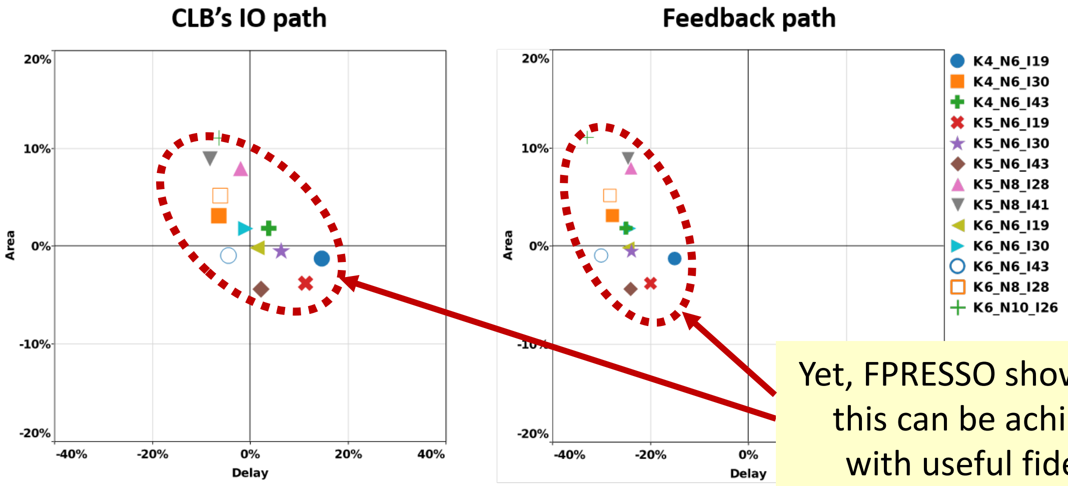
1 Exploring FPGA Architectures



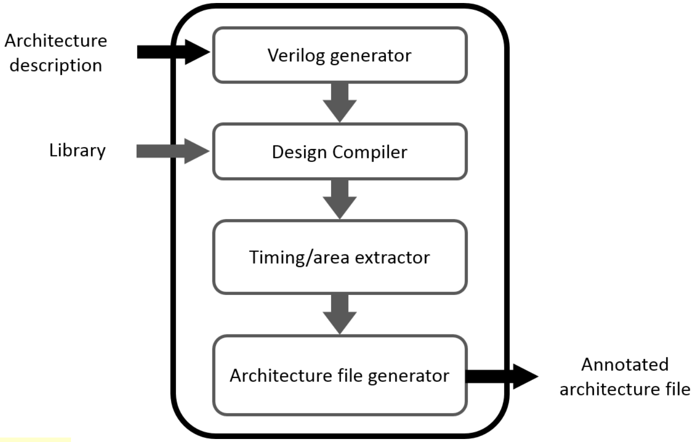
Problem: To study FPGA architectures, you need good retargetable FPGA toolsets (synthesis, mapping, P&R), such as VTR, but you also need good models of the transistor-level primitives (LUTs, crossbars, novel logic elements) and of their combinations

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

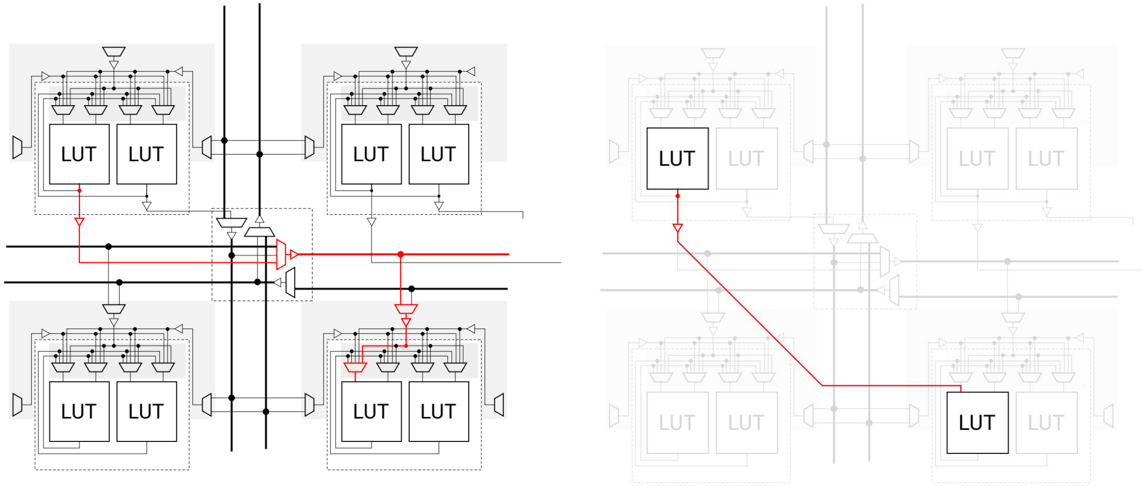


fpresso.epfl.ch

Zgheib, Lortkipanidze, Owaida, Novo, and Ienne. *FPRESSO: Enabling Express Transistor-level Exploration of FPGA Architectures*. FPGA 2016 **Best Paper Award**
Zgheib and Ienne. *Evaluating FPGA Clusters under Wide Ranges of Design Parameters*. FPL 2017 **Best Paper Award Nominee**

1 Efficient FPGA Interconnect

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?**

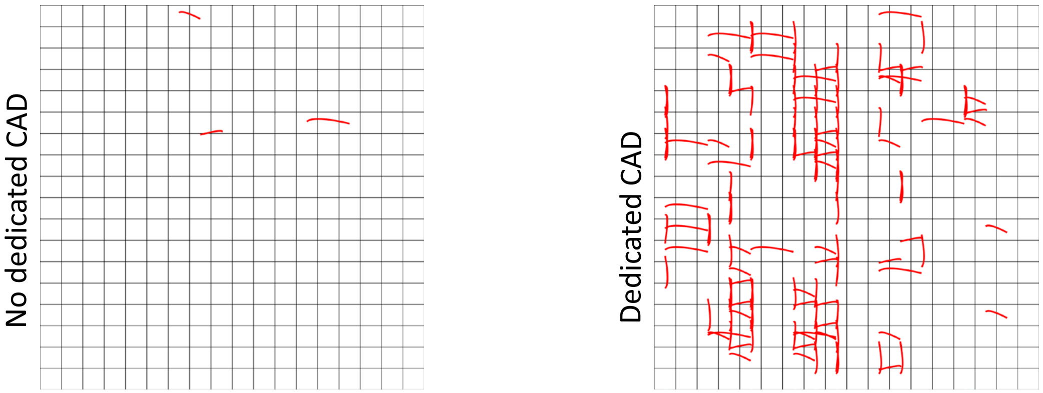


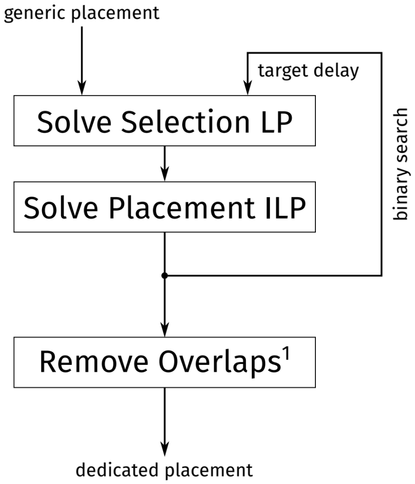
Idea: Introduce **fast and cheap direct connections**

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

Nikolić, Zgheib, and Ienne. *Straight to the Point: Intra- and Intercluster LUT Connections to Mitigate the Delay of Programmable Routing*. FPGA 2020

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





generic placement

target delay

Solve Selection LP

Solve Placement ILP

binary search

Remove Overlaps[1]

dedicated placement

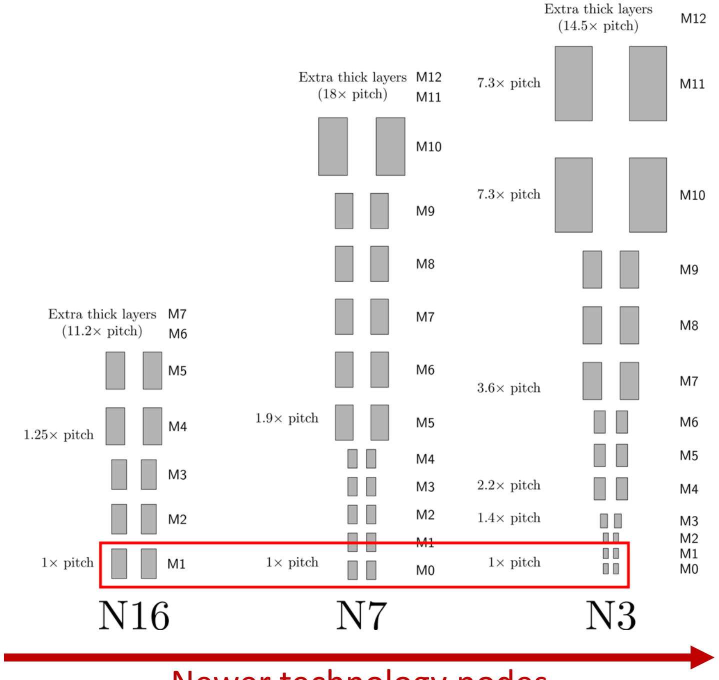
Nikolić, Zgheib, and Ienne. *Timing-Driven Placement for FPGA Architectures with Dedicated Routing Paths*. FPL 2020 **Michal Servit Best Paper Award**

1 FPGAs in Scaled Technologies

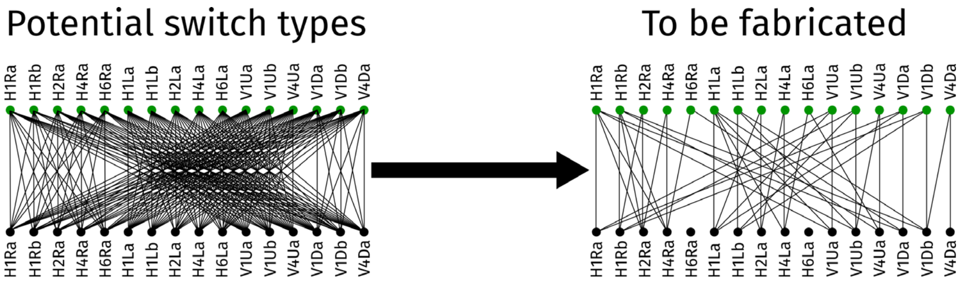
Problem: Interconnect delays scale poorly (R and C grow quickly and wires become “slower”)

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



Newer technology nodes


Potential switch types

To be fabricated

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

Nikolić, Catthoor, Tőkei, and Ienne. *Global is the New Local: FPGA Architecture at 5nm and Beyond*. FPGA 2021
Nikolić and Ienne. *Turning PathFinder Upside-Down: Exploring FPGA Switch-Blocks by Negotiating Switch Presence*. FPL 2021 **Michal Servit Best Paper Award**

2 Dynamically Scheduled HLS

dynamatic.epfl.ch

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)

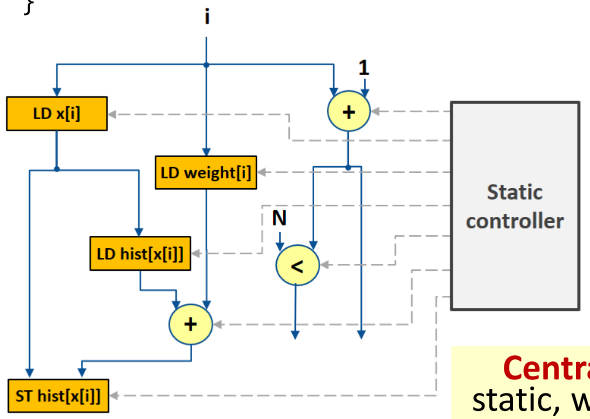
Idea: Create **dynamically scheduled** circuits where operations are executed as the operands are ready

- Control and data dependencies are resolved on-the-fly

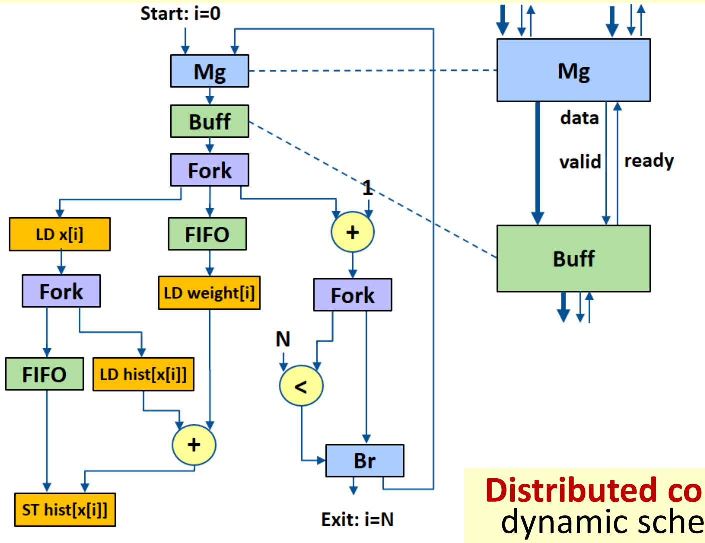
```
for (i=0; i<N; i++) {
   hist[x[i]] = hist[x[i]]
                + weight[i];
}
```

```
1: ld hist[5]; st hist[5];
2: ld hist[4]; st hist[4];
3: ld hist[4]; st hist[4];
```

RAW

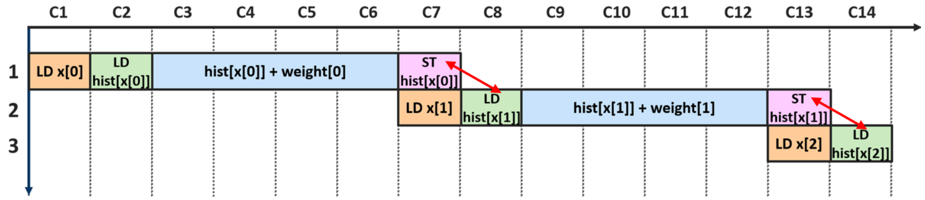


Centralized controller:
static, worst-case schedule

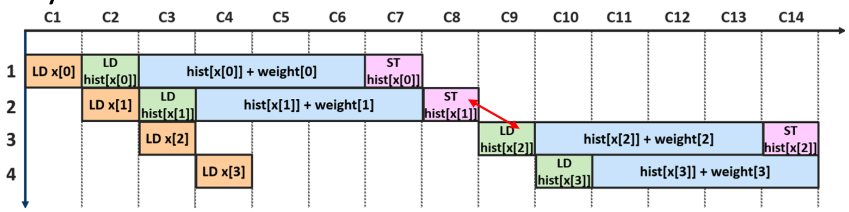


Distributed control:
dynamic schedule

Static HLS:



Dynamic HLS:

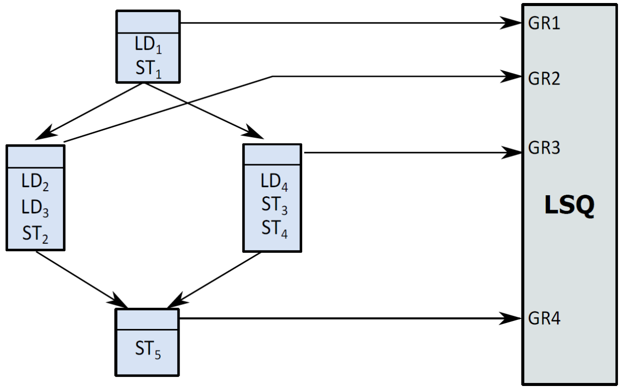


Josipović, Ghosal, and Ienne. *Dynamically Scheduled High-level Synthesis*. FPGA 2018 **Best Paper Award Nominee**
Josipović, Brisk, and Ienne. *From C to Elastic Circuits*. Asilomar SSC 2017
Josipović, Sheikhha, Guerrieri, Ienne, and Cortadella. *Buffer Placement and Sizing for High-Performance Dataflow Circuits*. FPGA 2020 **Best Paper Award**

2 An Out-of-Order Memory Interface for Spatial Computing

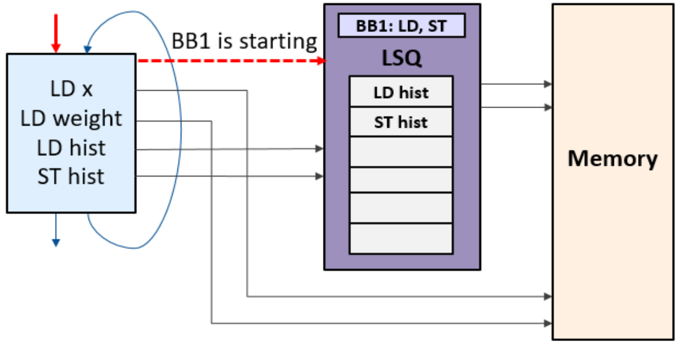
Problem: Conventional **Load-Store Queue allocation policy** is not suitable for spatial computing

- 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



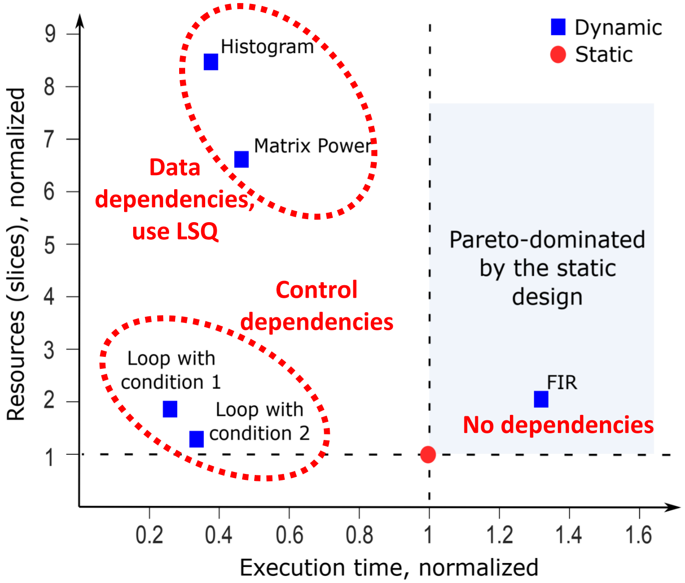
Idea: Allocate **groups of memory accesses** depending on the control flow of the application

- 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

Static vs. dynamic scheduling:



Josipović, Brisk, and Ienne. *An Out-of-Order Load-Store Queue for Spatial Computing*. CASES 2017 **Best Paper Award Nominee**

2 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

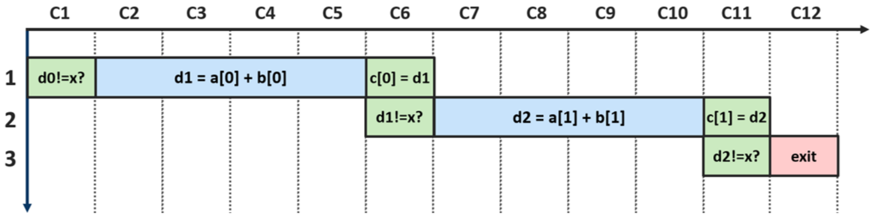
Idea: Contain **speculation in a region** of the circuit delimited by special components

- Issue speculative tokens (pieces of data which might or might not be correct)
- Squash and replay in case of misspeculation

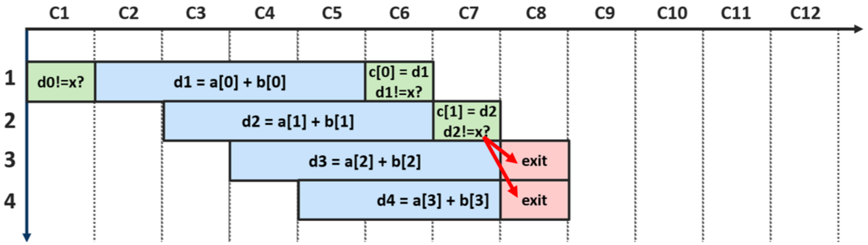
```
float d = 0.0; x = 100.0; int i = 0;

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
```

Nonspeculative schedule: conservatively wait for loop condition

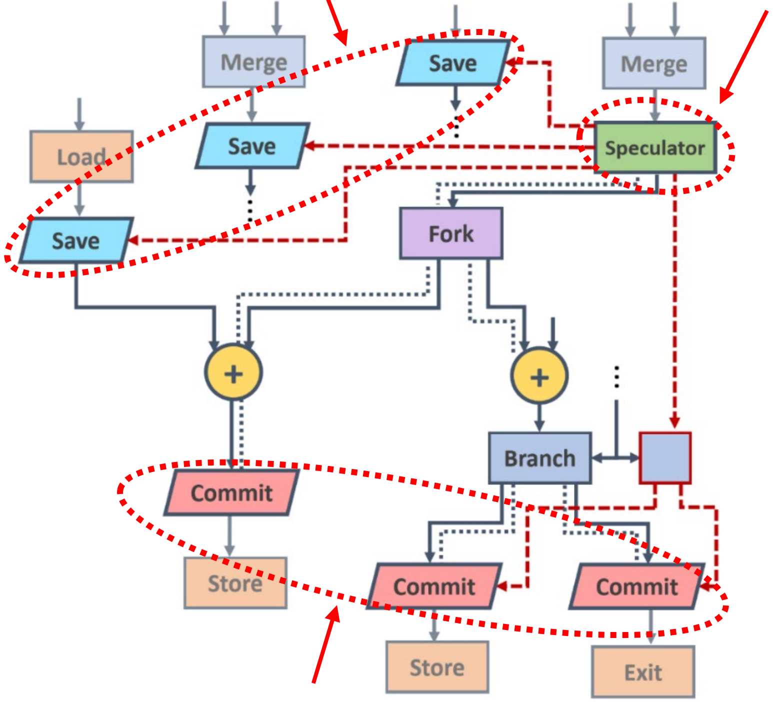


Speculative schedule: tentatively start another loop iteration



Save a copy of all regular tokens which may combine with a speculative token

Speculatively inject tokens into the speculative region

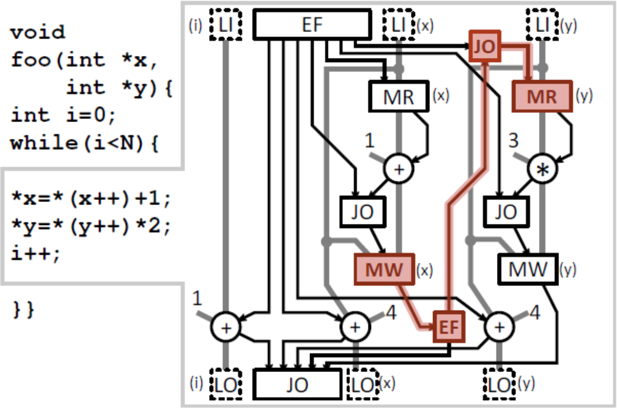


Commit only speculative results when they turn out to be correct

Josipović, Guerrieri, and Ienne. *Speculative Dataflow Circuits*. FPGA 2019

2 An Elastic Coarse Grain Reconfigurable Array

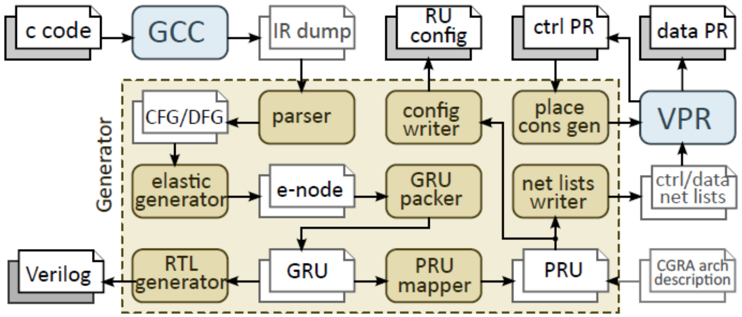
Elastic circuits mimic the dynamic processing of asynchronous circuits in a synchronous context and provide control-flow synchronization primitives (EB, EF, MG, JO, BR…)



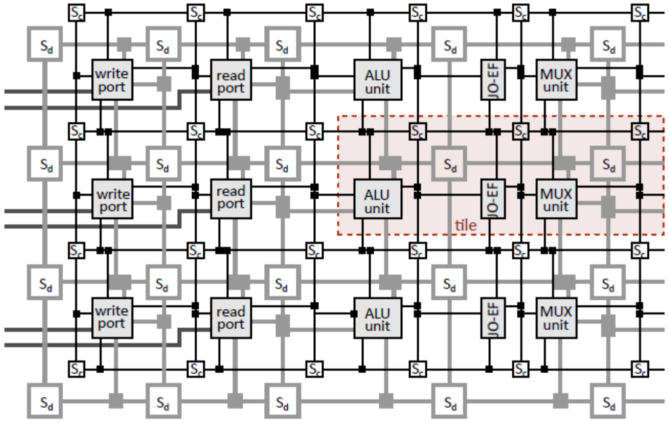
```
void
foo(int *x,
    int *y){
int i=0;
while(i<N){

*x=*(x++)+1;
*y=*(y++)*2;
i++;

}}
```

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!



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



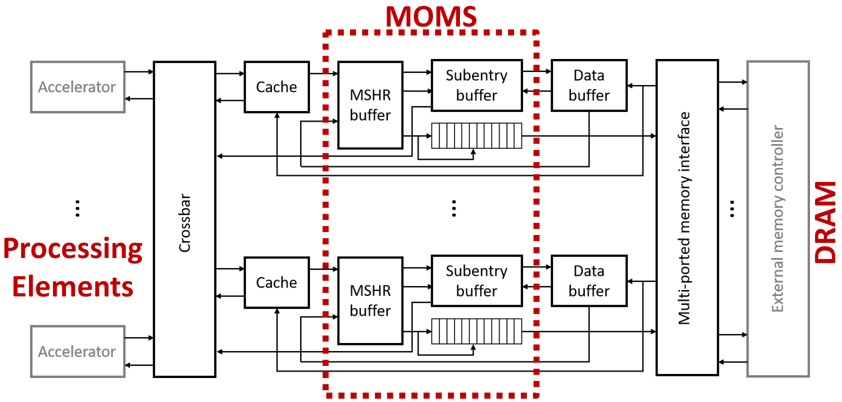
Najjar and Ienne. Special issue on *Reconfigurable Computing*. IEEE Micro (2014)

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

❸ Miss-Optimized Memory Systems (MOMSes)

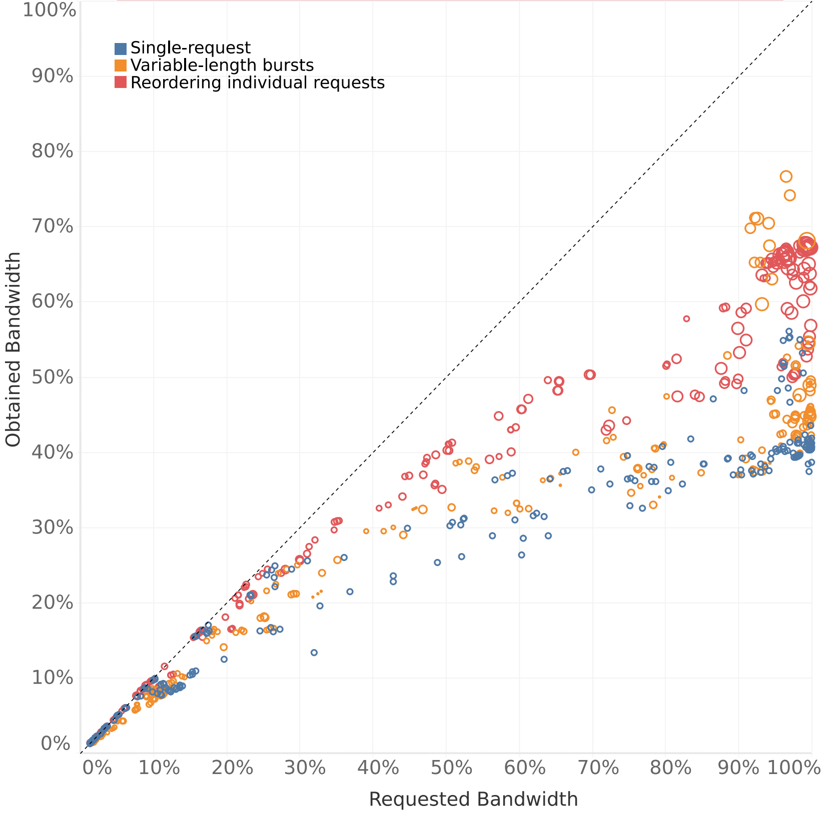
Problem: Effective DRAM bandwidth is more than one order of magnitude smaller than peak bandwidth when accesses from accelerators are irregular and narrow (e.g., sparse linear algebra, graph traversal)



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.

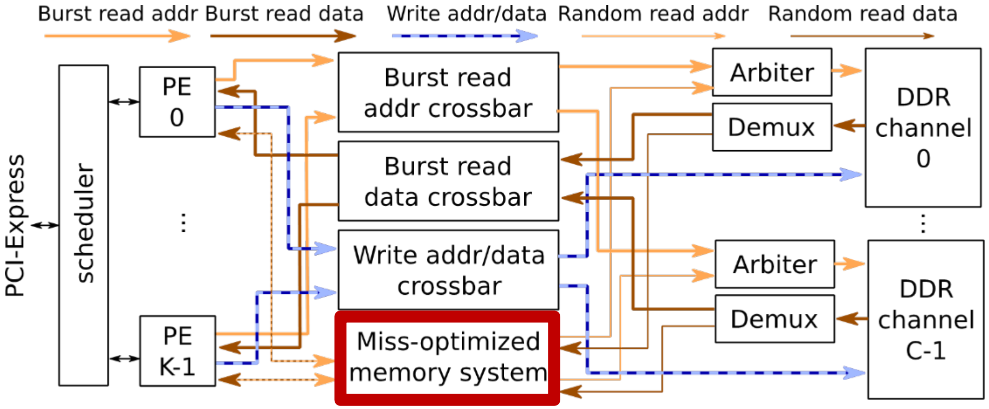
Challenge: How to implement efficiently the associative structure needed for 1000's of MSHRs?!

Wide exploration of design points:
90% of Pareto-optimal points are **MOMSes**
25% are MOMSes with **no cache!**



Csordás, Asiatici, and Ienne. *In Search of Lost Bandwidth: Extensive Reordering of DRAM Accesses on FPGA*. FPT 2019
Asiatici and Ienne. *DynaBurst: Dynamically Assemblying DRAM Bursts over a Multitude of Random Accesses.* FPL 2019
Asiatici and Ienne. *Stop Crying Over Your Cache Miss Rate: Handling Efficiently Thousands of Outstanding Misses in FPGAs*. FPGA 2019

4 Large-Scale Graph Processing on FPGAs with MOMS



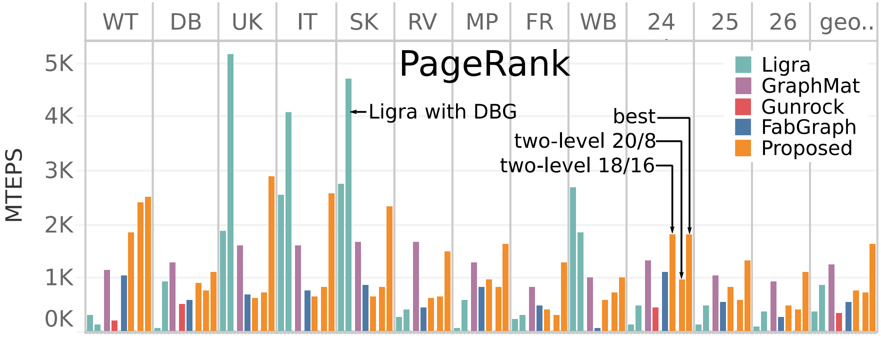
Problem: Show that MOMSes can be useful for **practical FPGA accelerators in the cloud**

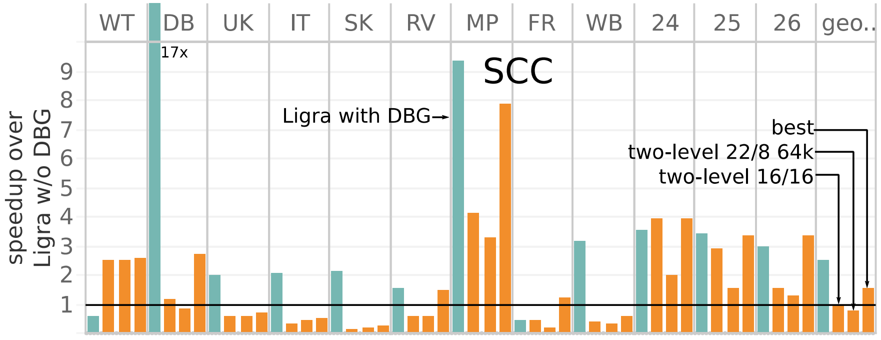
Intuition: No need for software defined data movement, dynamic data movement ("caches") can be made to work if designed appropriately

	Platform	Ext. mem. bandwidth	Power
This work, FabGraph	FPGA	64 GB/s	23 W
Gunrock	GPU	900 GB/s	300 W*
Ligra, GraphMat	CPU	233 GB/s	224 W

Results:

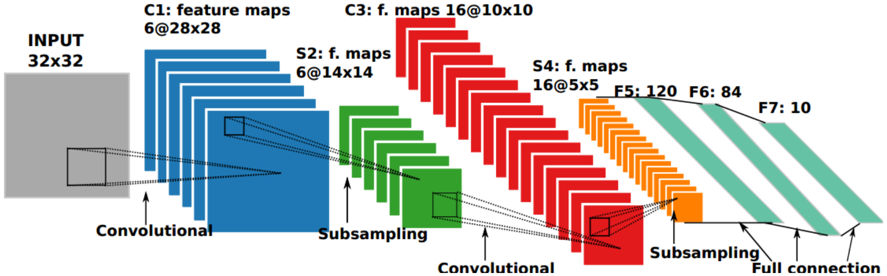
- PageRank:
  - 3x faster than state-of-the-art on FPGA
- PageRank, Strongly Connected Components:
  - competitive with state-of-the-art on CPU and GPU
  - 1.1x to 15.3x more bandwidth and power efficiency
- Single Source Shortest Path
  - competitive with SotA on CPU (→ higher bandwidth and power efficiency)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

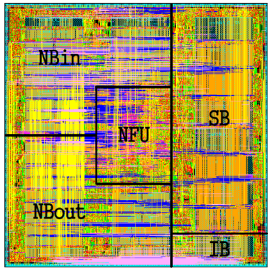




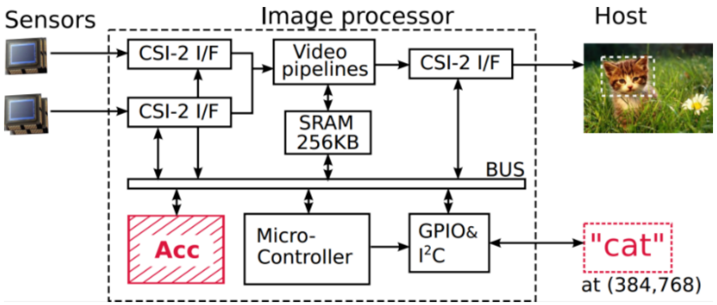
Asiatici and Ienne. *Large-Scale Graph Processing on FPGAs with Caches for Thousands of Simultaneous Misses.* ISCA 2021

4 Accelerating CNNs for Vision and Classification



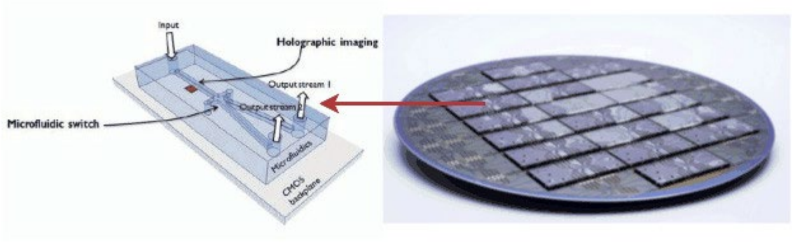


Problem: Convolutional and Deep Neural Networks (CNNs and DNNs) are extremely effective for some classification tasks but are terribly computationally demanding and poorly suited for GPGPUs



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)

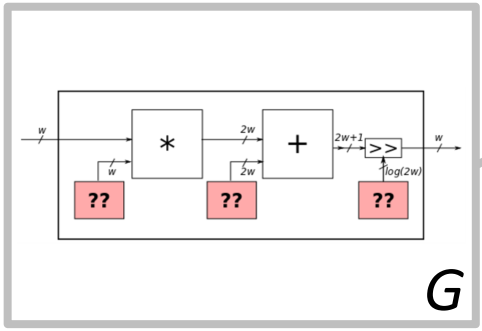
Du, Fasthuber, Chen, Ienne, Li, Luo, Feng, Chen, Temam. *ShiDianNao: Shifting Vision Processing Closer to the Sensor.* ISCA 2015

5 Improving Circuit Design with a "Fill-in-the-Blank" EDA Tool

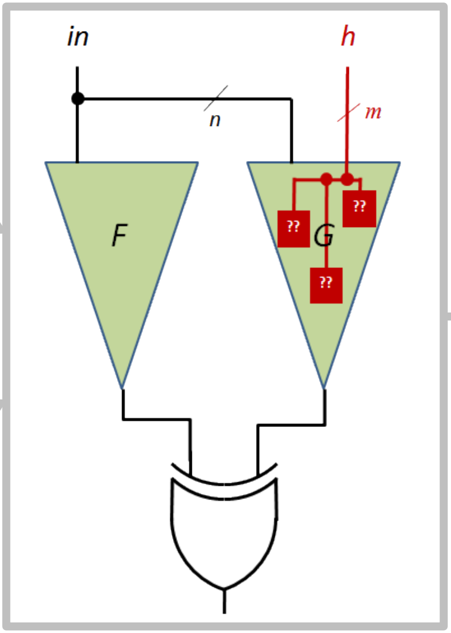
Problem: Humans are good at big-picture design, but not the details
Idea: Let software build the circuit from a designer's sketch

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

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

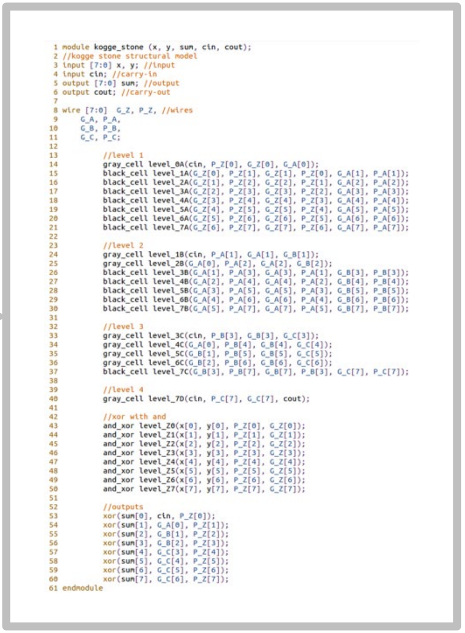
F



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



```
module kogge_stone (x, y, sum, cin, cout);
//kogge stone structural model
input [7:0] x, y; //input
input cin; //carry-in
output [7:0] sum; //output
output cout; //carry-out

wire [7:0]  G_Z, P_Z, //wires
    G_A, P_A,
    G_B, P_B,
    G_C, P_C;

        //level 1
        gray_cell level_0A(cin, P_Z[0], G_Z[0], G_A[0]);
        black_cell level_1A(G_Z[0], P_Z[1], G_Z[1], P_Z[0], G_A[1], P_A[1]);
        black_cell level_2A(G_Z[1], P_Z[2], G_Z[2], P_Z[1], G_A[2], P_A[2]);
        black_cell level_3A(G_Z[2], P_Z[3], G_Z[3], P_Z[2], G_A[3], P_A[3]);
        black_cell level_4A(G_Z[3], P_Z[4], G_Z[4], P_Z[3], G_A[4], P_A[4]);
        black_cell level_5A(G_Z[4], P_Z[5], G_Z[5], P_Z[4], G_A[5], P_A[5]);
        black_cell level_6A(G_Z[5], P_Z[6], G_Z[6], P_Z[5], G_A[6], P_A[6]);
        black_cell level_7A(G_Z[6], P_Z[7], G_Z[7], P_Z[6], G_A[7], P_A[7]);

        //level 2
        gray_cell level_1B(cin, P_A[1], G_A[1], G_B[1]);
        gray_cell level_2B(G_A[0], P_A[2], G_A[2], G_B[2]);
        black_cell level_3B(G_A[1], P_A[3], G_A[3], P_A[1], G_B[3], P_B[3]);
        black_cell level_4B(G_A[2], P_A[4], G_A[4], P_A[2], G_B[4], P_B[4]);
        black_cell level_5B(G_A[3], P_A[5], G_A[5], P_A[3], G_B[5], P_B[5]);
        black_cell level_6B(G_A[4], P_A[6], G_A[6], P_A[4], G_B[6], P_B[6]);
        black_cell level_7B(G_A[5], P_A[7], G_A[7], P_A[5], G_B[7], P_B[7]);

        //level 3
        gray_cell level_3C(cin, P_B[3], G_B[3], G_C[3]);
        gray_cell level_4C(G_A[0], P_B[4], G_B[4], G_C[4]);
        gray_cell level_5C(G_B[1], P_B[5], G_B[5], G_C[5]);
        gray_cell level_6C(G_B[2], P_B[6], G_B[6], G_C[6]);
        black_cell level_7C(G_B[3], P_B[7], G_B[7], P_B[3], G_C[7], P_C[7]);

        //level 4
        gray_cell level_7D(cin, P_C[7], G_C[7], cout);

        //xor with and
        and_xor level_Z0(x[0], y[0], P_Z[0], G_Z[0]);
        and_xor level_Z1(x[1], y[1], P_Z[1], G_Z[1]);
        and_xor level_Z2(x[2], y[2], P_Z[2], G_Z[2]);
        and_xor level_Z3(x[3], y[3], P_Z[3], G_Z[3]);
        and_xor level_Z4(x[4], y[4], P_Z[4], G_Z[4]);
        and_xor level_Z5(x[5], y[5], P_Z[5], G_Z[5]);
        and_xor level_Z6(x[6], y[6], P_Z[6], G_Z[6]);
        and_xor level_Z7(x[7], y[7], P_Z[7], G_Z[7]);

        //outputs
        xor(sum[0], cin, P_Z[0]);
        xor(sum[1], G_A[0], P_Z[1]);
        xor(sum[2], G_B[1], P_Z[2]);
        xor(sum[3], G_B[2], P_Z[3]);
        xor(sum[4], G_C[3], P_Z[4]);
        xor(sum[5], G_C[4], P_Z[5]);
        xor(sum[6], G_C[5], P_Z[6]);
        xor(sum[7], G_C[6], P_Z[7]);
endmodule
```

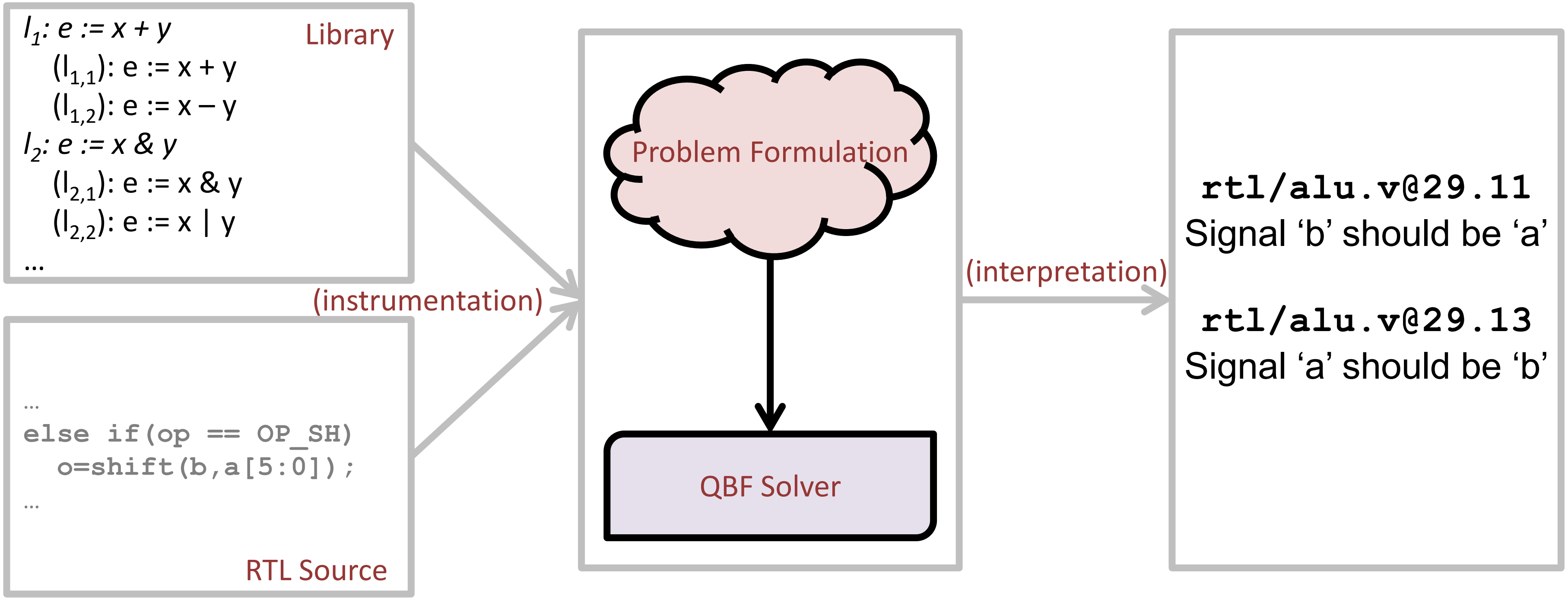
Designer gets back *completed*, optimized, guaranteed-correct circuit in RTL

Becker, Novo, and Ienne. *SKETCHILOG: Sketching combinational circuits.* DATE 2014
Becker, Novo, and Ienne. *Automated circuit elaboration from incomplete architectural descriptions.* Asilomar SSC 2013

5 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 | y
…

RTL Source

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

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

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

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

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

Becker, Maksimovic, Novo, Ewaida, Veneris, Jobstmann, and Ienne. *FudgeFactor: Syntax-Guided Synthesis for Accurate RTL Error Localization and Correction.* HVC 2015

6 SAT Methods for Logic Synthesis

	x_1	x_2	x_3	x_4	f
1	0	0	0	1	1
5	0	1	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
13	1	1	0	1	1

smallest SAT assignment

greatest SAT assignment

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

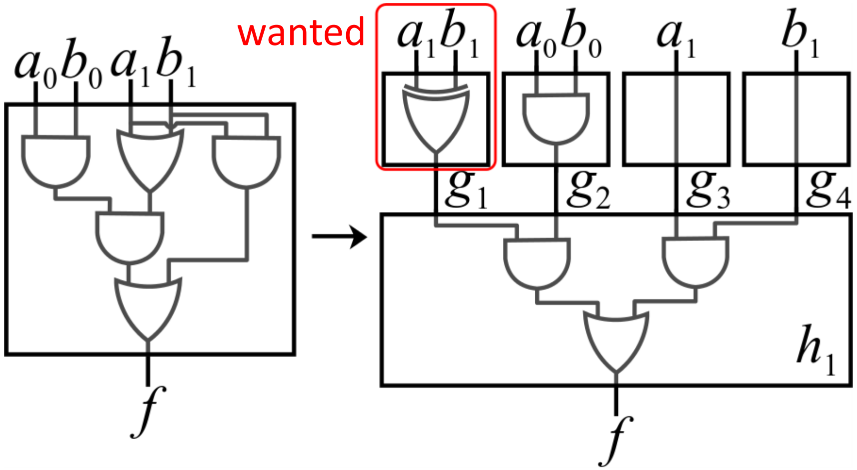
Petkovska, Mishchenko, Soeken, De Micheli, Brayton, and Ienne. *Fast Generation of Lexicographic Satisfiable Assignments: Enabling Canonicity in SAT-based Applications*. ICCAD 2016

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

6 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

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

Omits only 0.15% of the wanted base functions

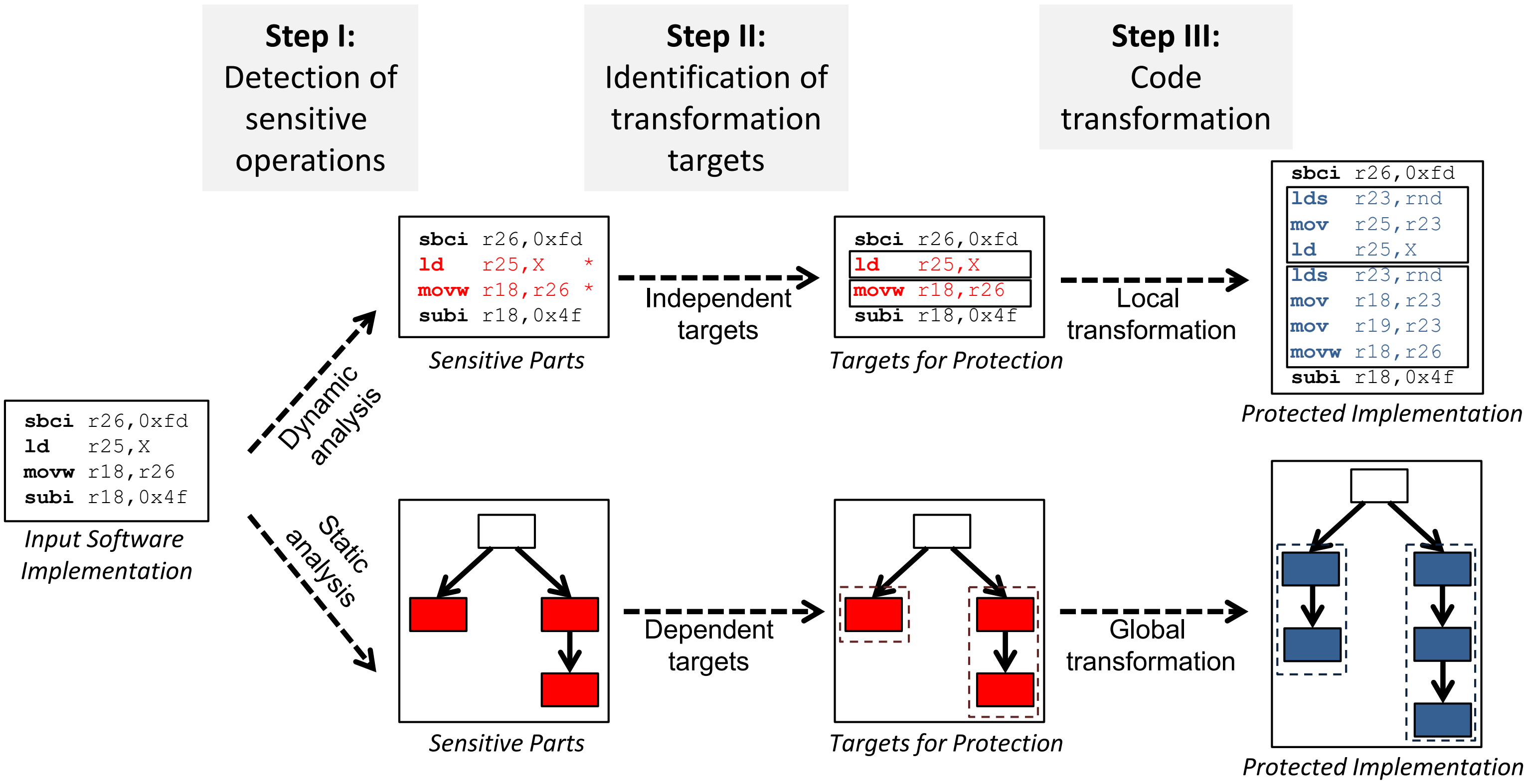
f is reconstructed using a Shannon expansion of two constrained interpolants built for $g_i = 1$ and $g_i = 0$, respectively

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**

7 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

Step II: Identification of transformation targets

Step III: Code transformation

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

Input Software Implementation

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

Sensitive Parts

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

Targets for Protection

```
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

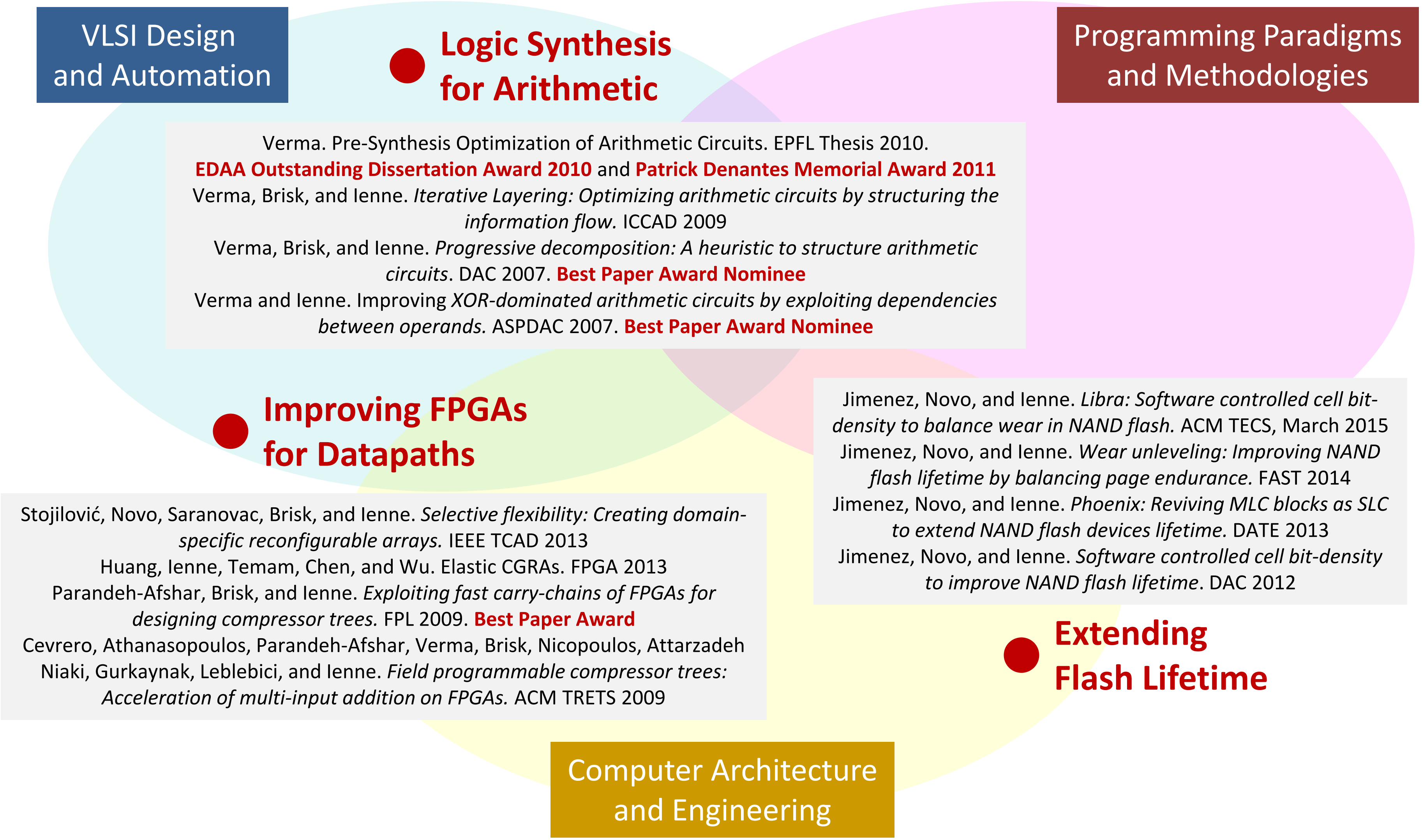
Sensitive Parts

Targets for Protection

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

Programming Paradigms and Methodologies

Computer Architecture and Engineering

● **Logic Synthesis for Arithmetic**

Verma. Pre-Synthesis Optimization of Arithmetic Circuits. EPFL Thesis 2010. **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, Brisk, and Ienne. *Progressive decomposition: A heuristic to structure arithmetic circuits*. DAC 2007. **Best Paper Award Nominee**

Verma and Ienne. Improving *XOR-dominated arithmetic circuits by exploiting dependencies between operands.* ASPDAC 2007. **Best Paper Award Nominee**

● **Improving FPGAs for Datapaths**

Stojilović, Novo, Saranovac, Brisk, and Ienne. *Selective flexibility: Creating domain-specific reconfigurable arrays.* IEEE TCAD 2013

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

Parandeh-Afshar, Brisk, and Ienne. *Exploiting fast carry-chains of FPGAs for designing compressor trees.* FPL 2009. **Best Paper Award**

Cevrero, Athanasopoulos, Parandeh-Afshar, Verma, Brisk, Nicopoulos, Attarzadeh Niaki, Gurkaynak, Leblebici, and Ienne. *Field programmable compressor trees: Acceleration of multi-input addition on FPGAs.* ACM TRETS 2009

Jimenez, Novo, and Ienne. *Libra: Software controlled cell bit-density to balance wear in NAND flash.* ACM TECS, March 2015

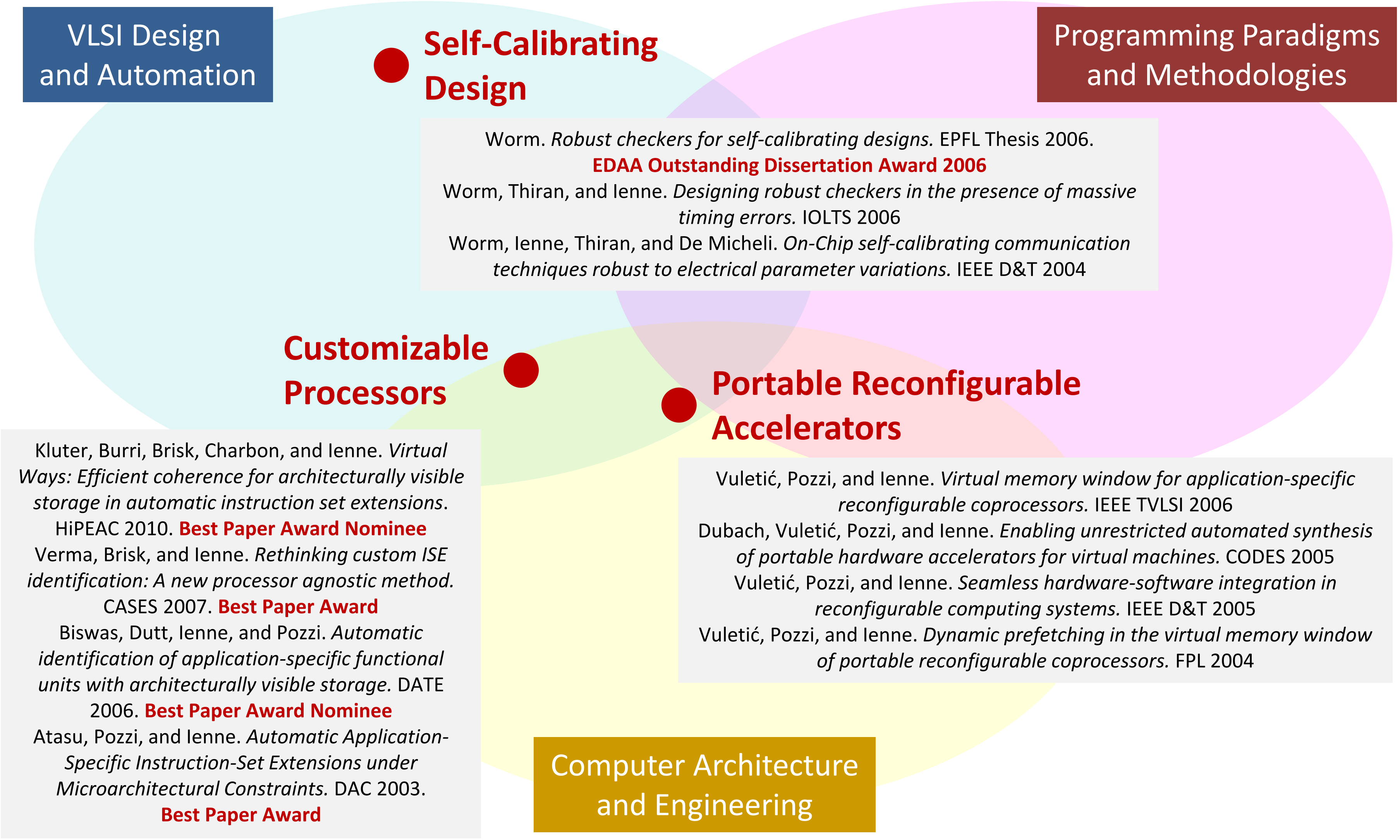
Jimenez, Novo, and Ienne. *Wear unleveling: Improving NAND flash lifetime by balancing page endurance.* FAST 2014

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

Jimenez, Novo, and Ienne. *Software controlled cell bit-density to improve NAND flash lifetime*. DAC 2012

● **Extending Flash Lifetime**

Selected Even Less Recent Topics



VLSI Design and Automation

Programming Paradigms and Methodologies

Computer Architecture and Engineering

Self-Calibrating Design

Worm. *Robust checkers for self-calibrating designs.* EPFL Thesis 2006. **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

Kluter, Burri, Brisk, Charbon, and Ienne. *Virtual Ways: Efficient coherence for architecturally visible storage in automatic instruction set extensions.* HiPEAC 2010. **Best Paper Award Nominee**
Verma, Brisk, and Ienne. *Rethinking custom ISE identification: A new processor agnostic method.* CASES 2007. **Best Paper Award**
Biswas, Dutt, Ienne, and Pozzi. *Automatic identification of application-specific functional units with architecturally visible storage.* DATE 2006. **Best Paper Award Nominee**
Atasu, Pozzi, and Ienne. *Automatic Application-Specific Instruction-Set Extensions under Microarchitectural Constraints.* DAC 2003. **Best Paper Award**

Portable Reconfigurable Accelerators

Vuletić, Pozzi, and Ienne. *Virtual memory window for application-specific reconfigurable coprocessors.* IEEE TVLSI 2006
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