LGraph: Live Graph infrastructure for synthesis and simulation

https://github.com/masc-ucsc/lgraph

Jose Renau

CSE Department
University of California, Santa Cruz
http://masc.soe.ucsc.edu

Micro Architecture Santa Cruz

UC SANTA CRUZ

My Background

- Professor at UC Santa Cruz
- Research area in computer architecture
- Consulting for high-end CPUs
  - 4-8 wide Out-of-order cores, coherence…
  - Large teams with unlimited synopsys licensing
  - Work on verification
  - Work with synthesis flow (DC/ICC/ICC2/PrimeTime…)

My Problem

- My problem

Hardware Design Productivity **is horrible**

- My target

Simulation and Synthesis (ASIC/FPGA)

- My hammers

Live Flow (1-30 secs response time)

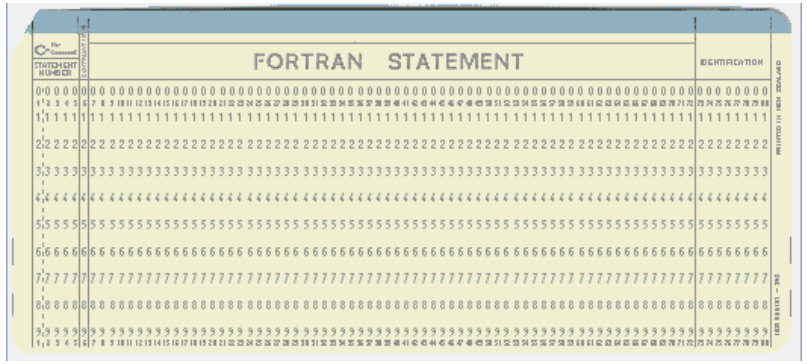
Synchronous and elastic pipelines

Enable new HDLs

“Live” opposite of “Batch”

- Live
  - Maximize “developer” utilization
  - Response under few seconds
  - Change code while running

- Batch
  - Maximize “computer” utilization
  - Submit job and check “hours” later for results



Why we need Live?

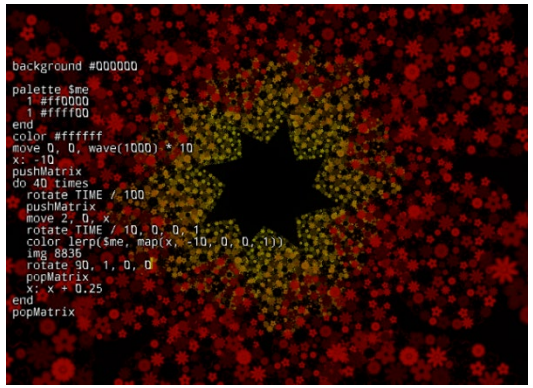
Human mnemonic memory
Typically 10-15 seconds

Initial delays over 2 secs have impact in Quality of Experience (QoE)

Breaks over a few seconds, require to “rebuild” memory

Who is going Live with computers?

- Music and Visual effects: Live Coding
  - Fast response to code changes
  - Music parties



- Teaching Programming
  - Interactive browser, programming

- Some Programming Languages
  - REPL == Read Eval Print Loop

Live Hardware Flow

Few seconds from code change to result
(no approx models)

- Simulation

Fast & Hot Reload

- FPGA

Place&route, bitstream

- ASIC

Timing, power, area feedback

LGraph

https://github.com/masc-ucsc/lgraph

masc-ucsc / lgraph

Code | Issues 35 | Pull requests 0 | Projects

Live Graph infrastructure for Synthesis and Simulation

live synthesis fpga asic simulation hdl lgraph Mana

1,349 commits 1 branch

Some Open Source Tools for HW

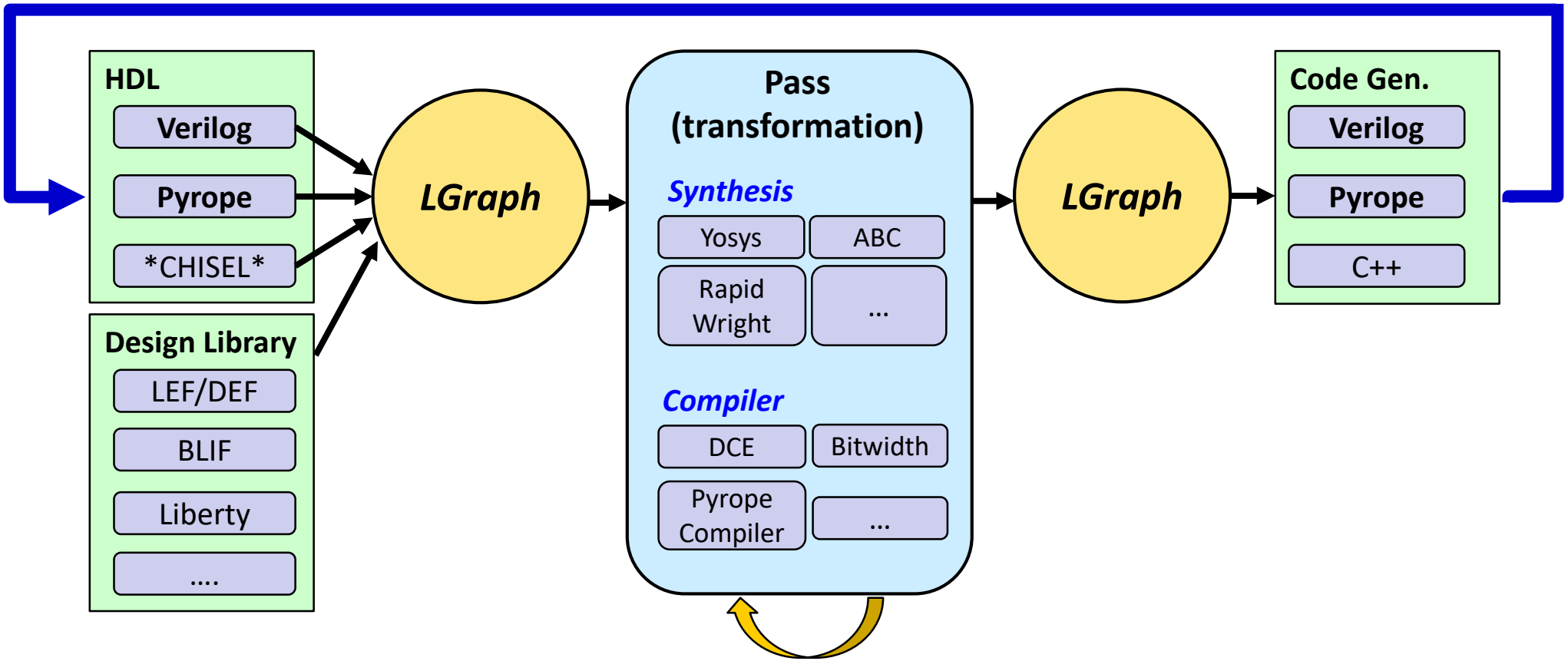
HDL	Front End	Back End
PyMTL	Verilator	Ophidian
CHISEL	Yosys	Rsyn
Pyrope	ABC	Arachne-pnr
		Next-pnr
		VTR

LGraph, why not “x tool”?

- Commercial tools
  - I need source to change the internals (incremental, load/save/…)
- Rsyn-x, https://github.com/RsynTeam/rsyn-x
- FIRRTL, https://github.com/freechipsproject/firrtl
- Yosys, https://github.com/YosysHQ/yosys
  - I want synthesis AND simulation
  - Significant rework to get Live structures
  - No focus on debug

LGraph: An Unified Infrastructure

- Role: the Hardware LLVM (openaccess++ but open, ndm++…)

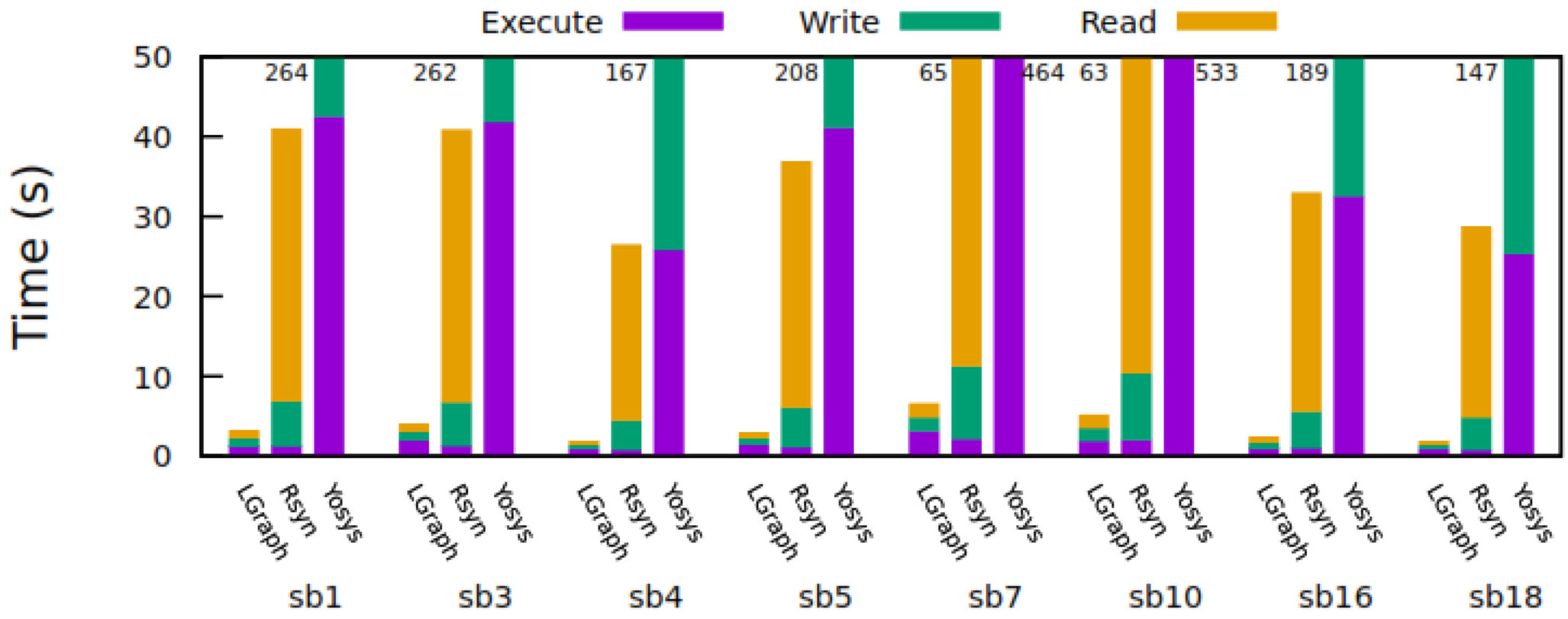


UCSC Components on Live Hardware

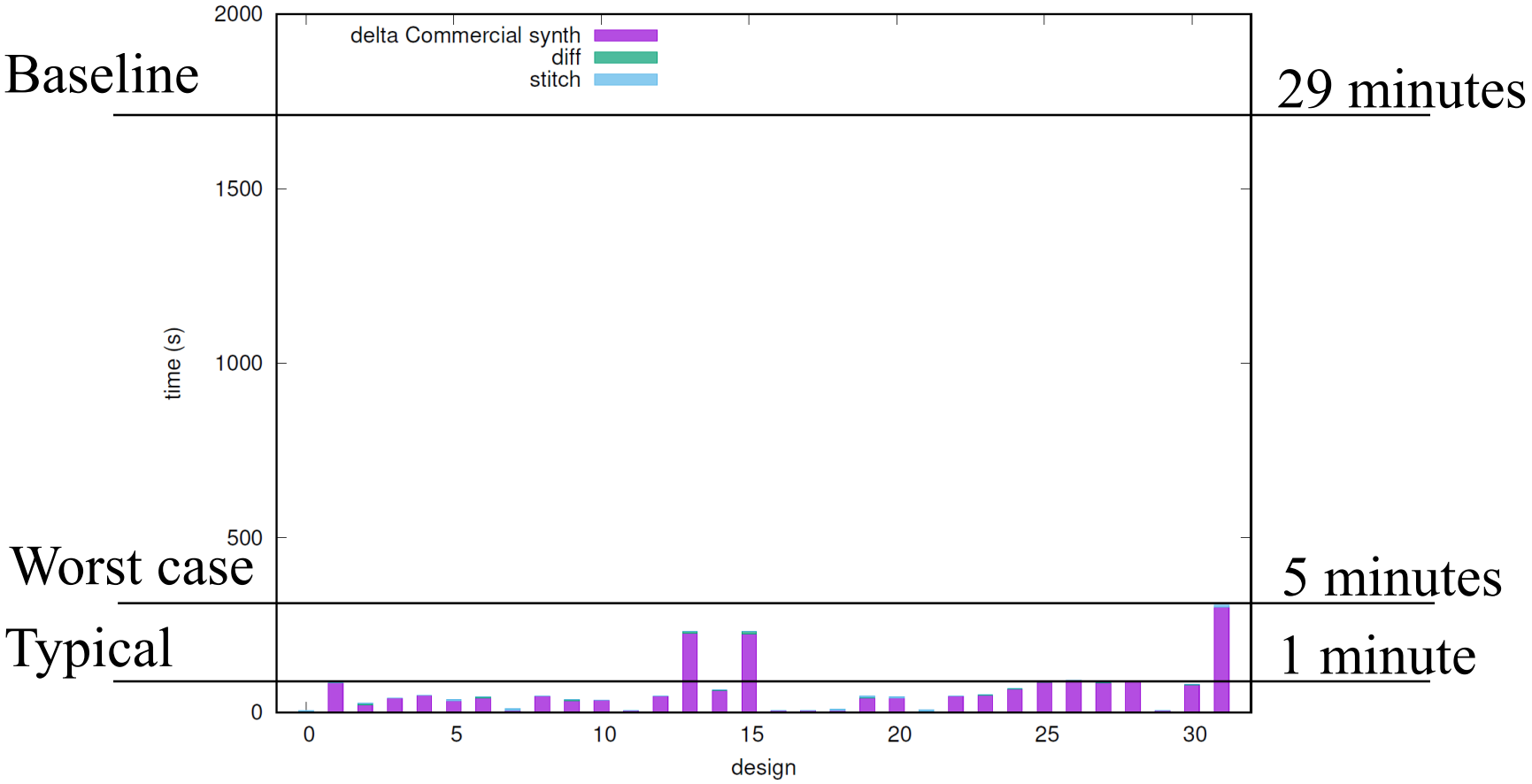
- LGraph https://github.com/masc-ucsc/lgraph
  - LLVM-like internal compiler for hardware with a Live focus
  - Simulation, FPGA, and ASIC target
  - Built to support new HDLs like Pyrope
    - https://masc.soe.ucsc.edu/pyrope.html

Some Results

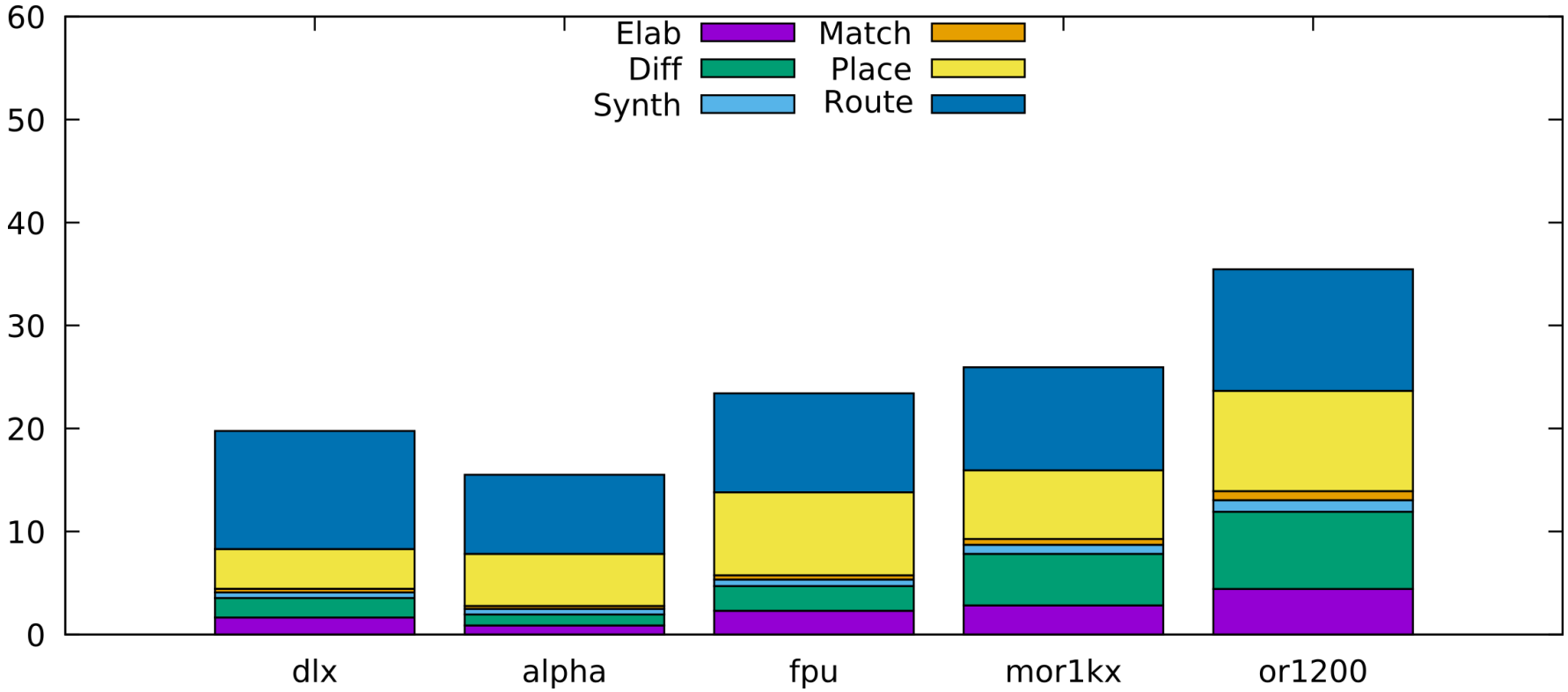
LGraph is Fast!



ASIC (DC) Incremental Synthesis under 60 secs

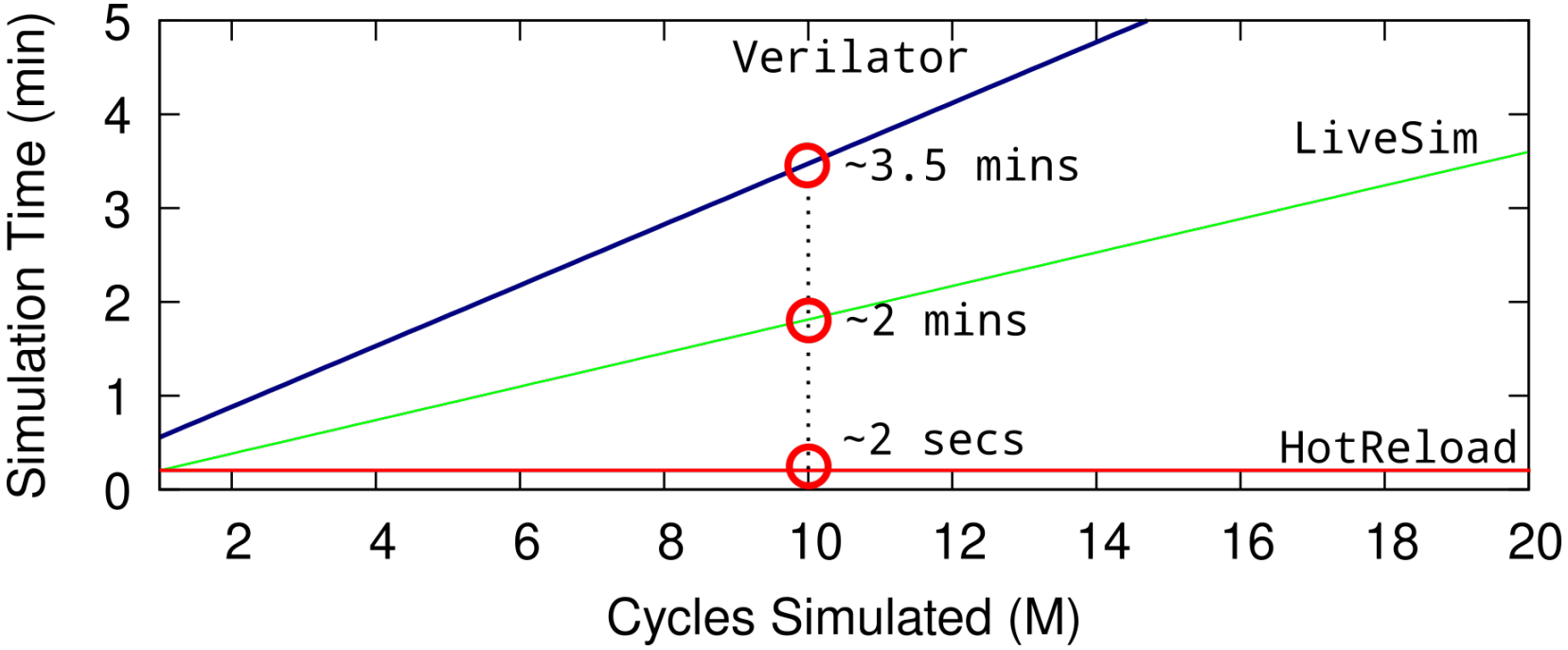


Live FPGA


60
50
40
30
20
10
0
Elab
Diff
Synth
Match
Place
Route
dlx
alpha
fpu
mor1kx
or1200

Live Simulation with Hot Reload

(prototype still not released at github)



LGraph Contributors

Past

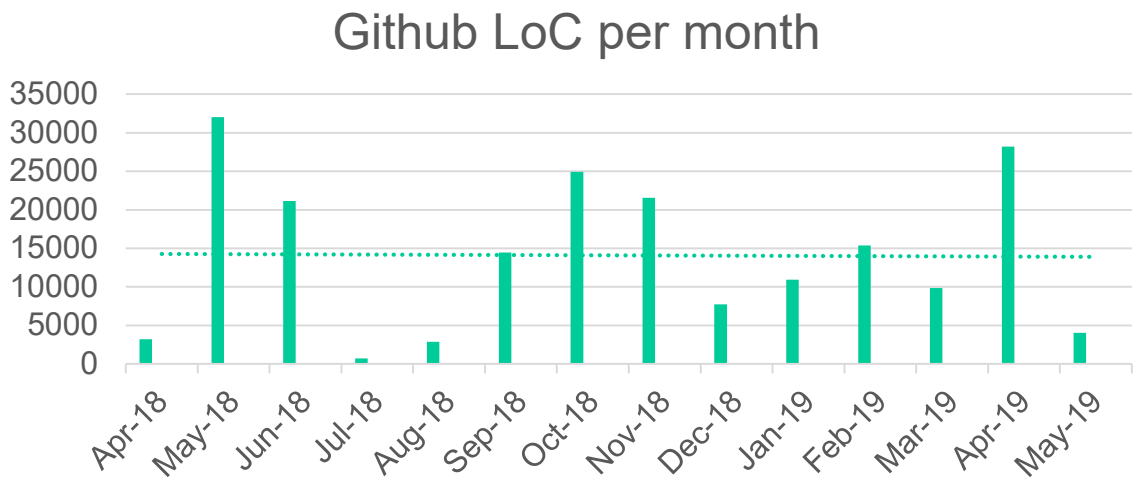
- PhD
  - Rafael T. Possignolo
  - Haven Skinner
- MS
  - Yuxun Qiu
  - Garvit Mantri
  - Yuxiong Zhu
- Undergraduates
  - Zachary Potter

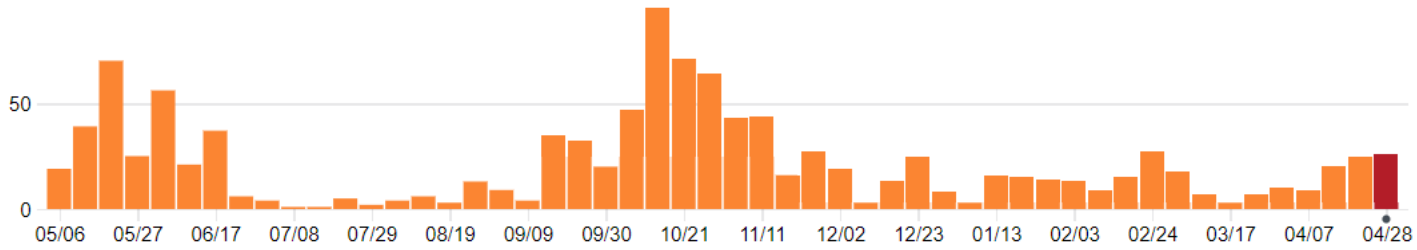
Current

- PhD
  - Sheng Hong Wang (Verifiable Compiler), 2H2020?
  - Nursultan Kabylkas (Code coverage), 1H2020?
- MS
  - Rohan Ganpati (OpenTimer), 2H2019
  - Qian Chen (Mockturtle) 2H2019
  - Rohan Jobanputra (Cloud Setup) 2H2019
  - Kenneth Meyer (Pyrope Parser), 1H2020?
  - Joshua Pena (Verilog and C++ code generation) 1H2020
  - Huijie Pan (RapidWright) 2H2020
  - Brian Metz (Verilog Parser) 2H2020
  - Hunter Coffman (Bitwise Compilation)
  - Some other MS that are just starting

LGraph is a significant effort

- Around +100K LoC changes per Year





Tool	LoC
Ophidian	9K C++
TaskFlow	13K C++
OpenTimer	15K C++
Qrouter	18K C
RapidWright (Xilinx)	24K Java
Netgen (LVDS)	32K C
LGraph	**36K C++17**
Rsyn-x (including GUI)	52K C++
Yosys	78K C++
Graywolf (placer)	82K C
ABC (no vudd, zlib…)	488K C

Graphviz Output Pass Example

- 105 LoC
  - 23 Header
  - 82 Code

```
std::string data = "digraph {\n";

g->each_node_fast([&data](const Node &node) {
  auto node_name = node.has_name() ? node.get_name() : "";

  data += fmt::format(" {} [label=\"{} :{} :{}\"];\n"
                      , node.debug_name(), node.debug_name(), node.get_type().get_name(), node_name);

  for (auto &out : node.out_edges()) {
    auto &dpin       = out.driver;
    auto  dnode_name = dpin.get_node().debug_name();
    auto  snode_name = out.sink.get_node().debug_name();
    auto  dpin_name  = dpin.has_name() ? dpin.get_name() : "";
    auto  bits       = dpin.get_bits();

    data += fmt::format(" {}->{}[label=\"{}b :{} :{} :{}\"];\n"
                        , dnode_name, snode_name, bits, dpin.get_pid(), out.sink.get_pid(), dpin_name);
  }
});

g->each_graph_output([&data](const Node_pin &pin) {
  std::string_view dst_str = "virtual_dst_module";
  auto             bits    = pin.get_bits();
  data += fmt::format(" {}->{}[label=\"{}b\"];\n", pin.get_node().debug_name(), dst_str, bits);
});

data += "}\n";
```

Current LGraph is 0.1 (alpha)

LGraph 0.2 goal (tried Latchup, but missed)

- New API based on feedback from users
  - Get back yosys, ABC, json, Incremental…

Code quality: codefactor A | code quality A | coverage 52%

Short CI: build failing | Long CI: Azure Pipelines failed

- New instance/type annotations heavily inspired by Adam FIRRTL work
  - https://www.youtube.com/watch?v=4YGIdjMNI6Q

- OpenTimer integration
  - https://github.com/OpenTimer/OpenTimer

- Mockturtle (ABC alternative) integration
  - https://github.com/lsils/mockturtle

LGraph 0.3 Goals (End of Year)

- New AST sub-project (fast incremental elaboration)
  - To/From LGraph
  - Custom code generation: C++, Verilog, Pyrope
  - Custom Pyrope parser
- Annotations with incremental
- Incremental parsing + synthesis BOOM in under 2 seconds
- Rapidwright integration
- Code coverage for simulation, support for finding bugs

DEMO

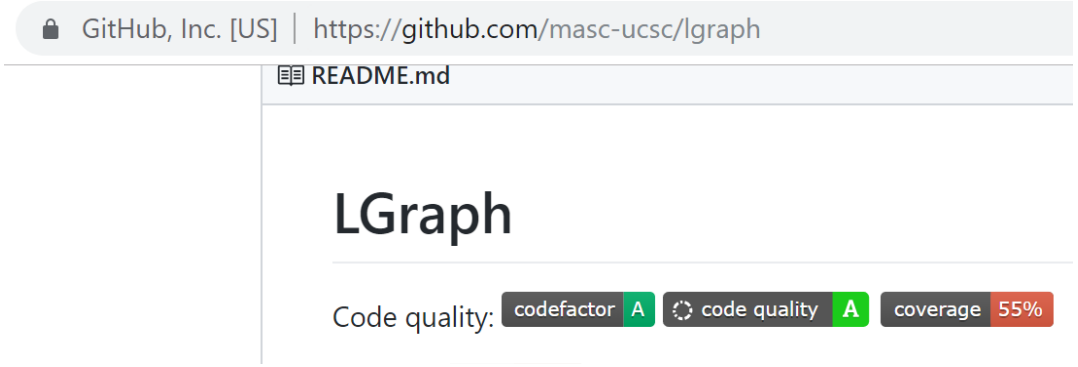
Jose Renau

renau@ucsc.edu

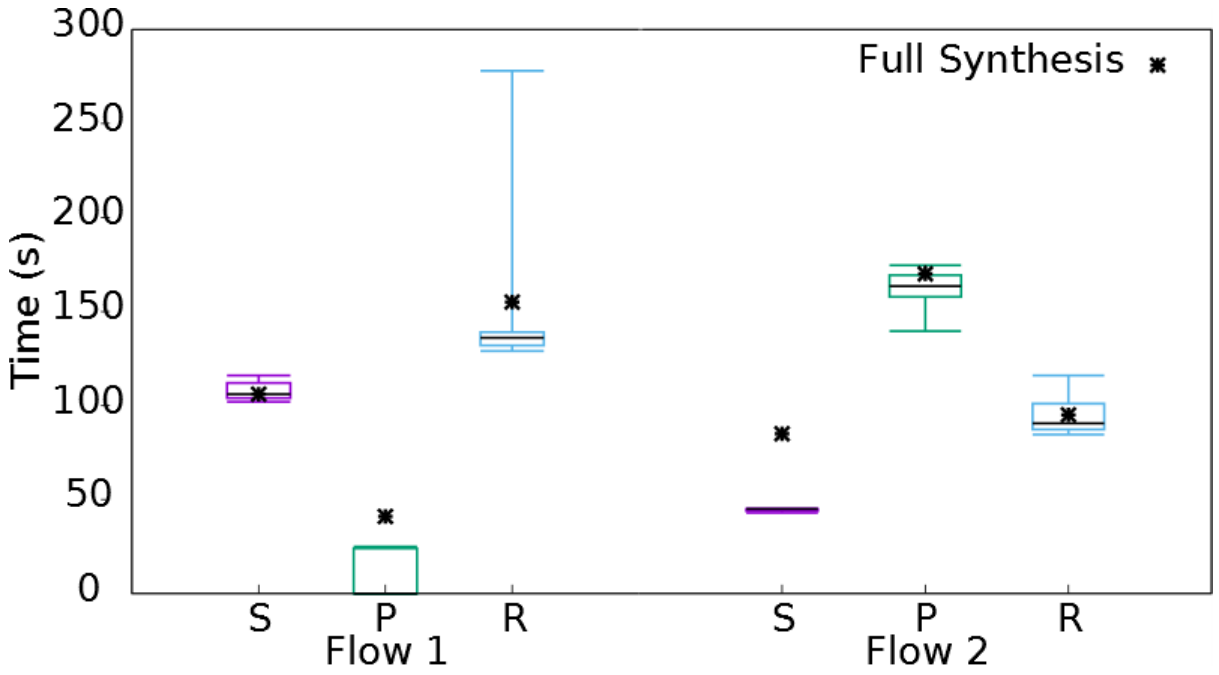
Department of Computer Science Engineering, University of California, Santa Cruz

http://masc.soe.ucsc.edu

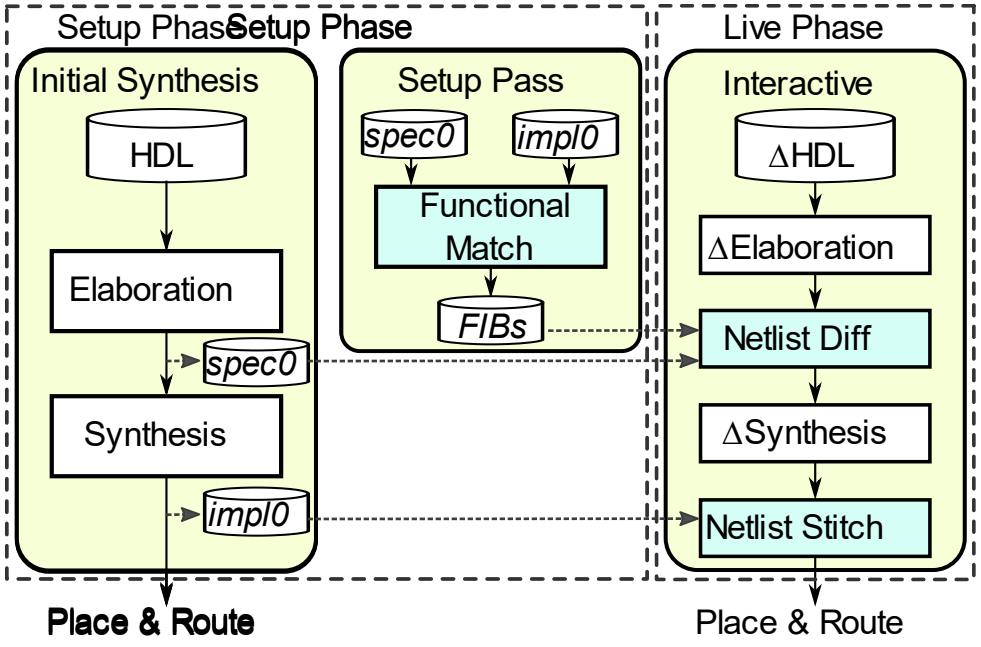
https://github.com/masc-ucsc/lgraph



Current FPGA flows for a “no change”

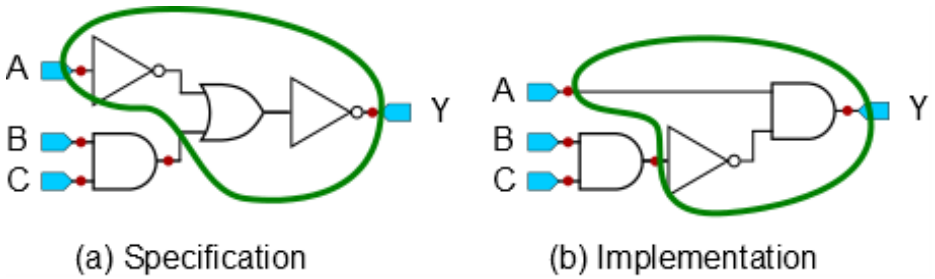


Live ASIC/FPGA: Flow Overview



•LiveSynth: Towards an Interactive Synthesis Flow, Rafael T. Possignolo, and Jose Renau, Design Automation Conference (**DAC**), June 2017.

How big are the changes?



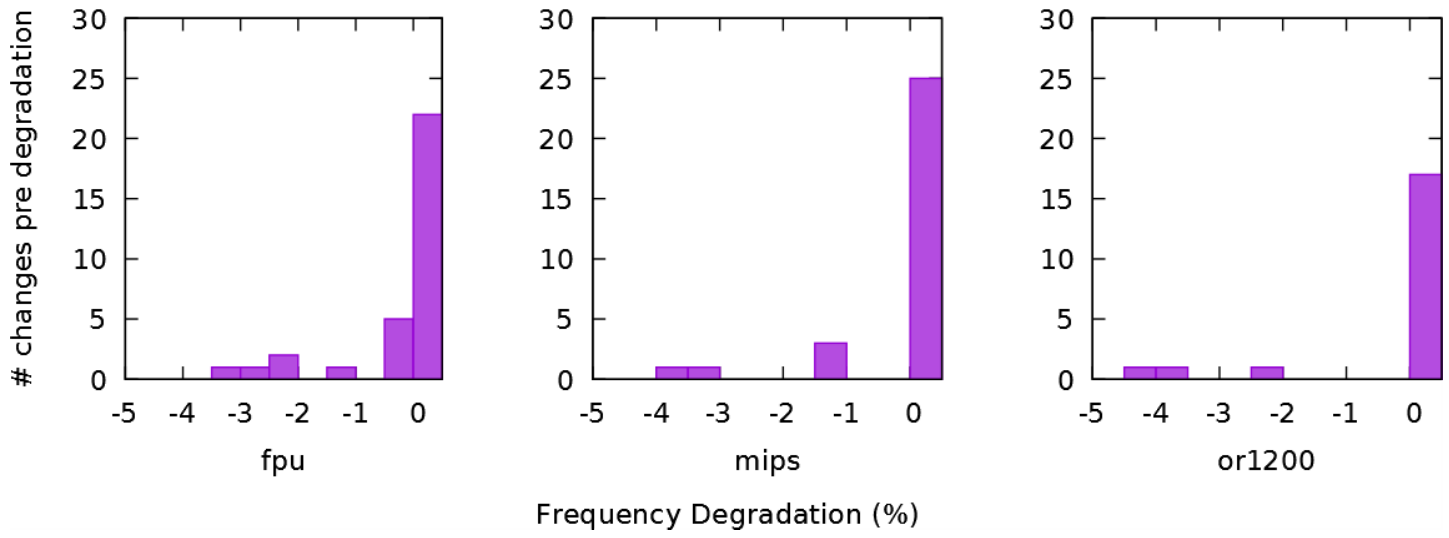
(a) Specification (b) Implementation

Functionally Invariant cones

- No change during synthesis
- Can be plugged in and out without any effort

Cone Size	fpu	mips	or1200
<200	1769	1237	643
200-300	99	73	172
300-400	938	35	156
400-500	1	2	185
500-600	649	11	74
600-800	34	316	63
800-1000	33	29	58
1000-1500	5	124	56
1500-2000	1	550	0
2000-3000	0	421	0
3000-4000	0	302	0
>4000	0	115	0

QoR degradation



- In most cases there is less than 1% difference when compared to full synthesis
- The maximum observed difference was ~4%

Fluid Pipelines

- Fluid Pipelines https://github.com/masc-ucsc/fluid
  - A new elastic pipeline methodology to have composable transformations

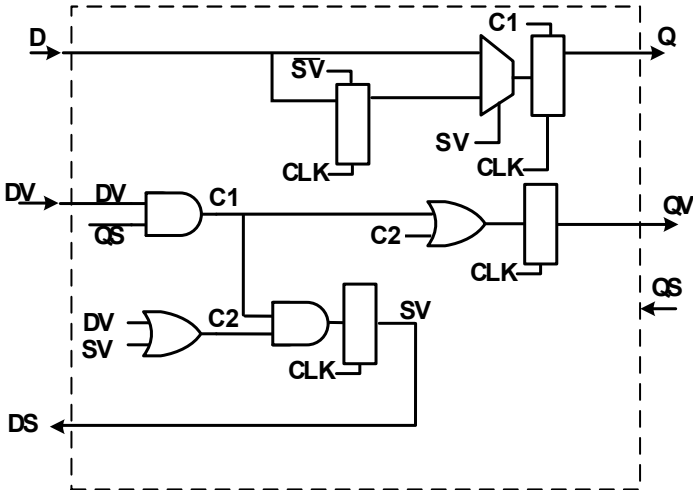
•Liam: An Actor Based Programming Model for HDLs, Haven Skinner, Rafael T. Possignolo, and Jose Renau. 15th ACM-IEEE International Conference on Formal Methods and Models for System Design (**MEMOCODE**), October 2017.

•Fluid Pipelines: Elastic Circuitry meets Out-of-Order Execution, Rafael T. Possignolo, Elnaz Ebrahimi, Haven Skinner, and Jose Renau, International Conference on Computer Design (**ICCD**), June 2016.

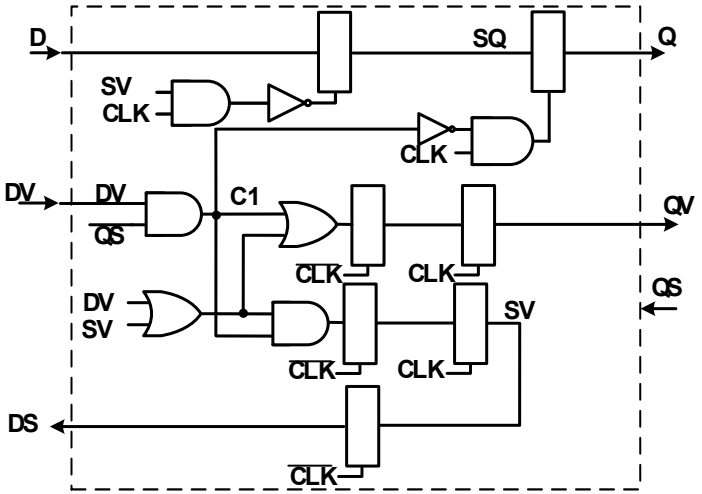
•Fluid Pipelines: Elasticity without Throughput Penalty, Rafael T. Possignolo, Elnaz Ebrahimi, Haven Skinner, and Jose Renau, International Workshop on Logic and Synthesis (**IWLS**), April 2016.

Fluid Flop (aka Elastic Buffer or Relay or..)

- Traditional flop is a “1 element FIFO”
- Fluid Flop is a 2 element FIFO with latches or flops



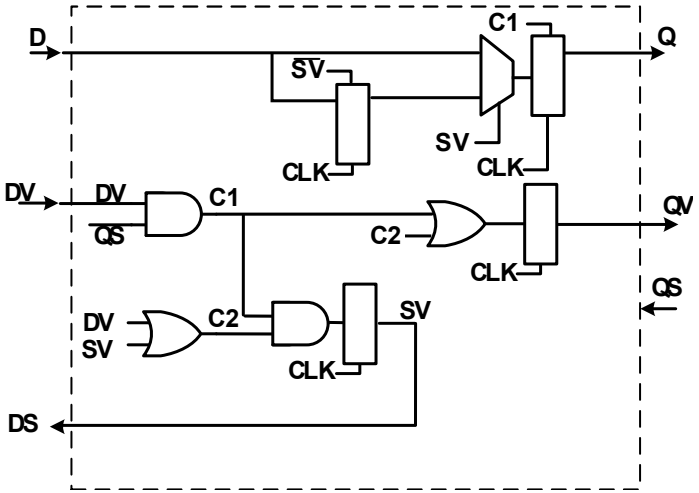
Flop based implementation



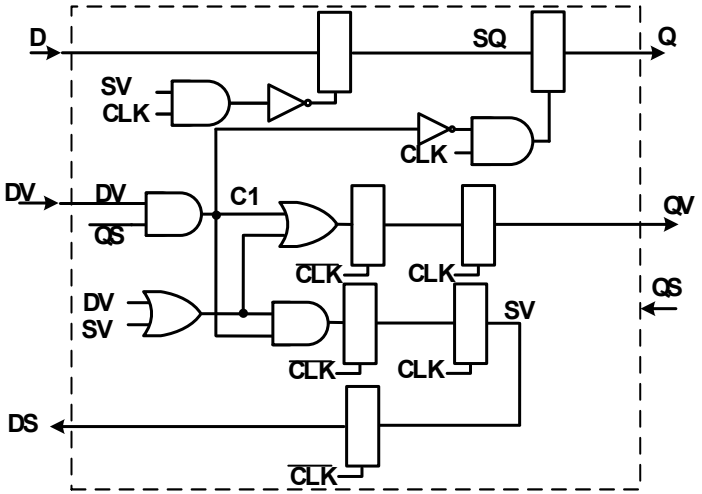
Latch based implementation

Fluid Flop (aka Elastic Buffer or Relay or..)

- Traditional flop is a “1 element FIFO”
- Fluid Flop is a 2 element FIFO with latches or flops



Flop based implementation



Latch based implementation

Fluid Pipelines

- Ideal for FPGAs
  - Expand the LUT to have Fluid Handshake with low cost
  - Allow tool to do aggressive "local" re-pipelining

Live Projects

- LGraph
  - Live synthesis
  - Live FPGA Bitstream
  - Live Simulation
  - RapidWright integration for FPGA flow
  - OpenTimer integration
  - Custom/Incremental Verilog parser
  - …

- Pyrope, a new HDL https://masc.soe.ucsc.edu/pyrope.html