Live Hardware Development at UCSC

Jose Renau
CSE Department
University of California, Santa Cruz

Pyrope

https://masc.soe.ucsc.edu/pyrope.html

LiveHD

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

Background

- Jose Renau
  - Professor at UC Santa Cruz
  - Simulation and traditional architecture, out-of-order design and verification
  - Consulting for high-end CPUs

- MASC Team working on LiveHD
  - Sheng-Hong Wang, Akash Sridhar, Sakshi Garg, Arik Yueh, Maximillian Tiao, Kyle Neil

- Graduated students
  - Haven Skinner, Hunter Coffman, Rafael T. Possignolo, Kenneth Mayer…

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

What if houses were build like CHIPS?

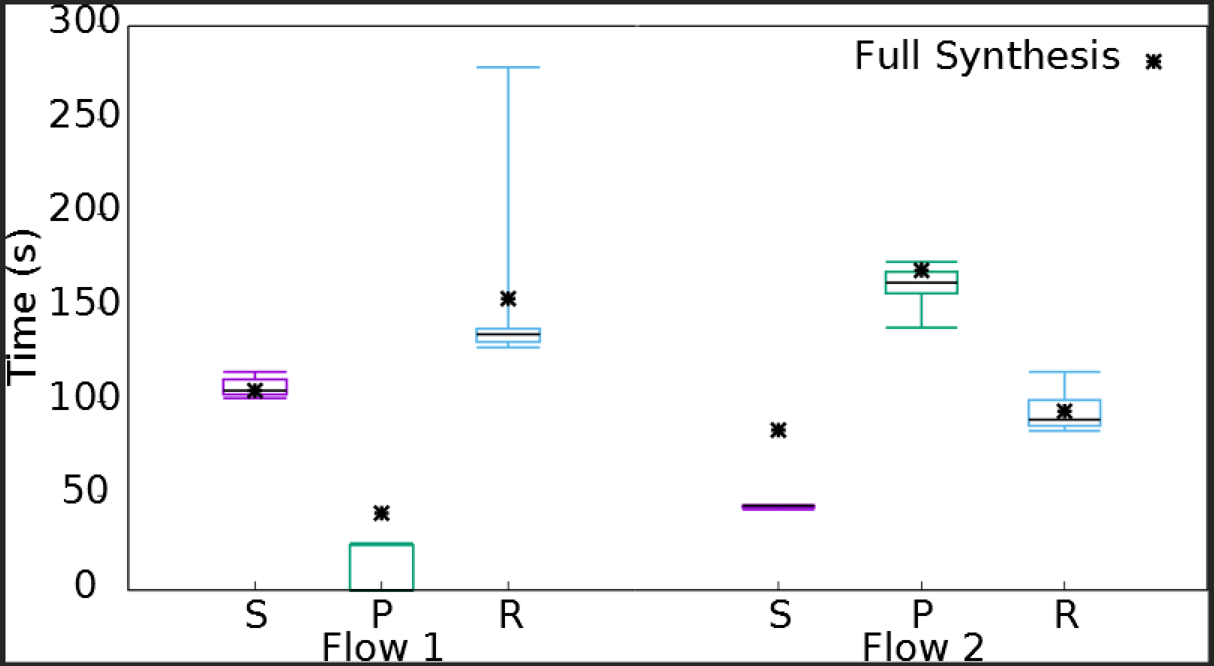
- Imagine you want to change one door size

What if houses were build like CHIPS?

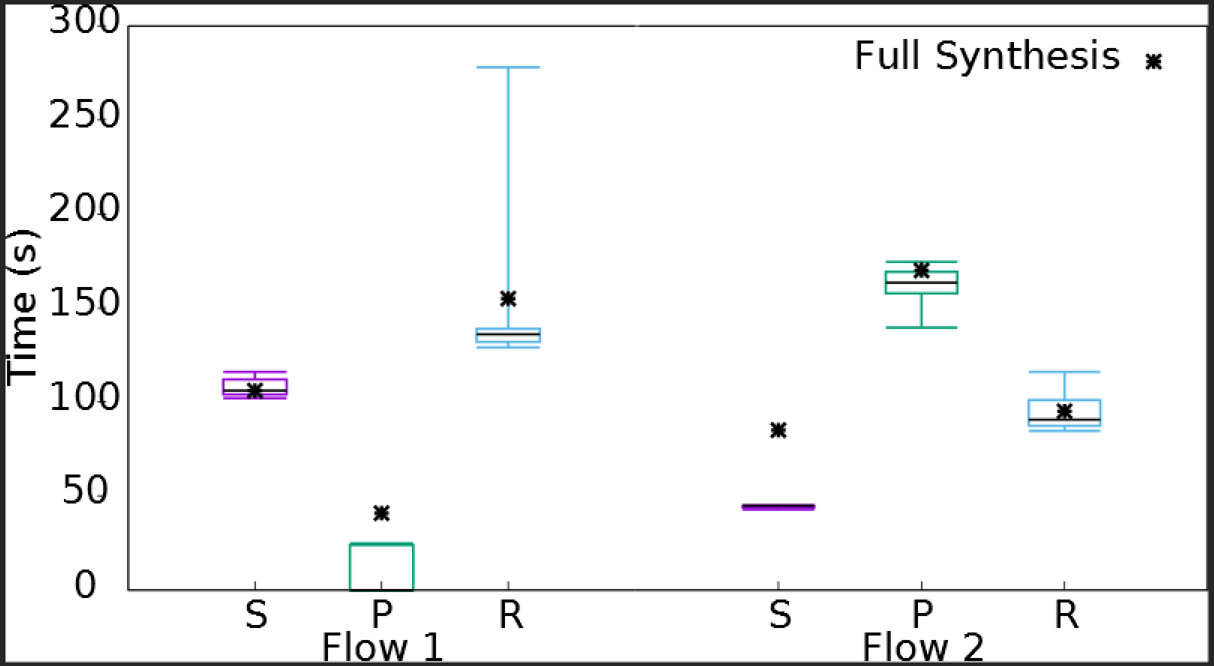
- Imagine you want to change one door size

make clean

Current FPGA flows for a “no change”

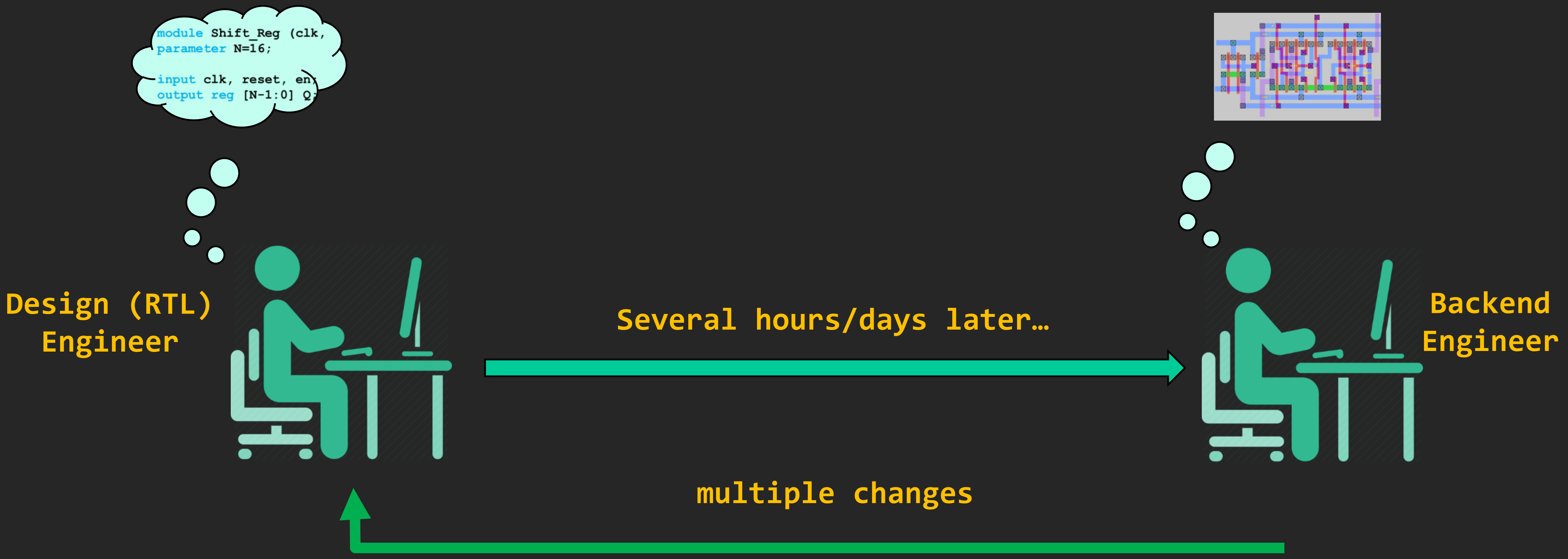


Current FPGA flows for a “no change”



Less than 20% time reduction!!!

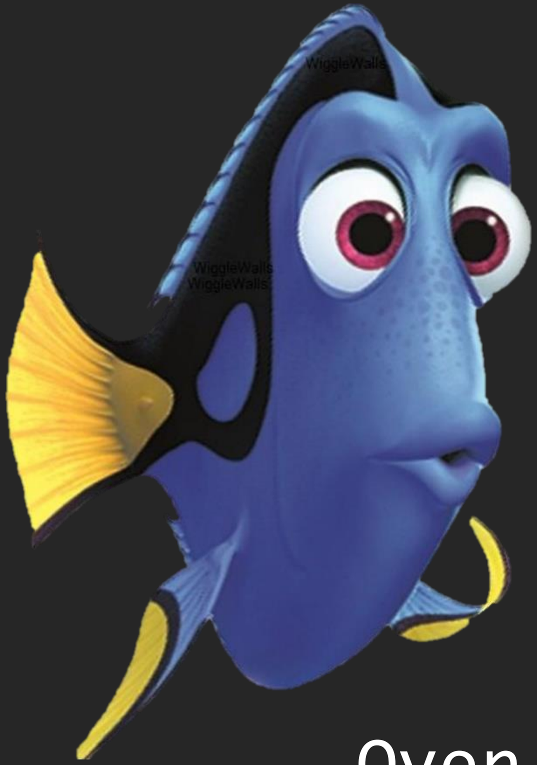
Current Chip Fabrication Flow



```
module Shift_Reg (clk,
parameter N=16;

input clk, reset, en
output reg [N-1:0] Q
```

Worse! We are Dori when solving problems

Few second interruptions require to “rebuild” memory



Human mnemonic memory is 10-15 seconds

Over 2 secs have impact in quality of experience

How it should be?

- Incremental
  - No “make clean”
- Very fast feedback
  - Propagate frontend and backend feedback in seconds (Dori)
- Modern Hardware Description Language (HDL)
  - Verilog (1984) is the de-facto industry standard

Why we started a new HDL called Pyrope

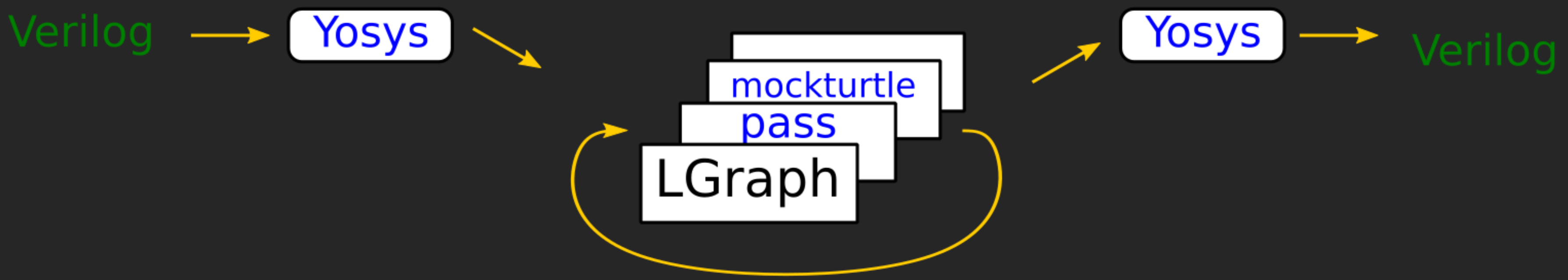
- In 2006, we started to build an out-of-order core
  - “The tools” demonstrated to be a problem
  - Very long iterations for edit/debug/verify
  - Disconnection from frontend and backend hardware design
  - Verilog did not allow to parametrize design
  - …

- In 2008, OoO was cancelled, and moved to a new language
  - “Pyrope Alpha” effort lead by UCSC PhD Haven Skinner
- Pyrope is “a stone that resembles a ruby”

By 2016, we realized that language is part of the solution

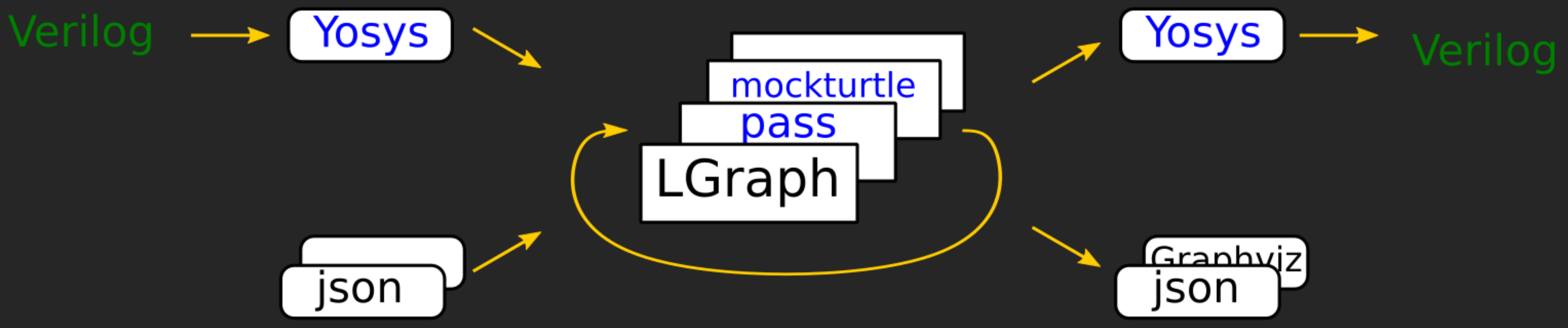
- The language syntax is only part of the problem
- The flow is more important than the language
  - Live simulation, reload, synthesis, scalable
  - Language constructs should integrate with the flow
    - Fluid constructs to give freedom to the flow
    - The language must interact with the flow

Live Hardware Development (LiveHD)



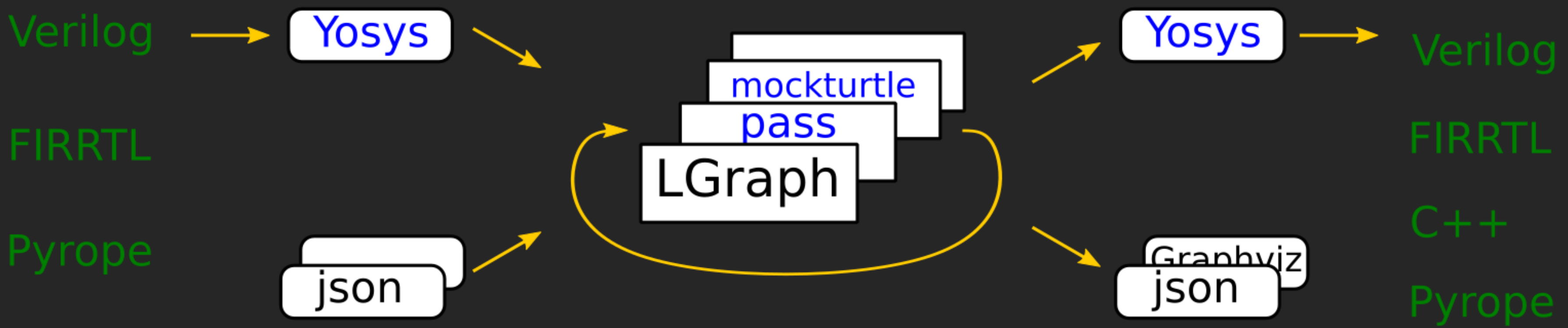
LLVM-like flow for incremental scalable hardware design

Live Hardware Development (LiveHD)



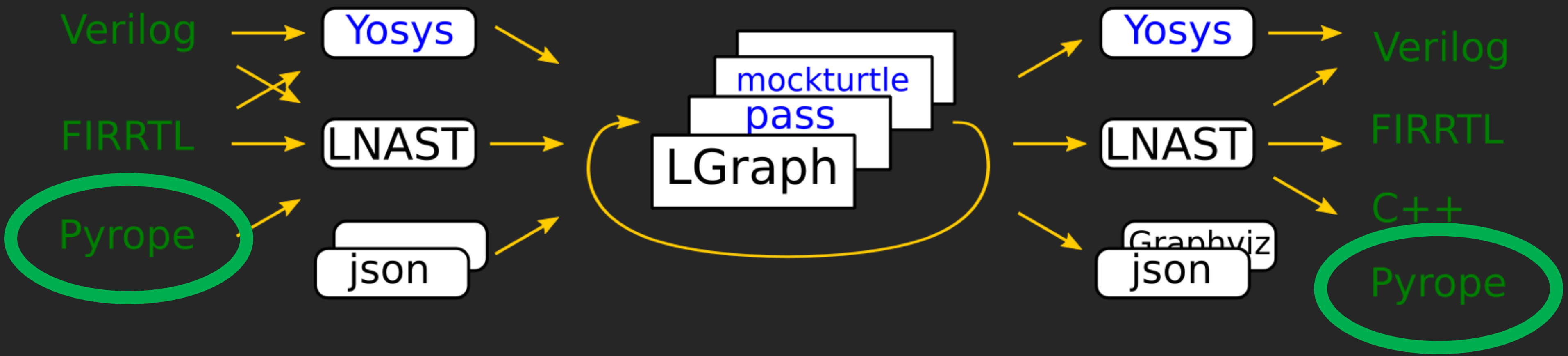
Many passes can be added

Live Hardware Development (LiveHD)



Fully synthesizable languages as input/output

Live Hardware Development (LiveHD)

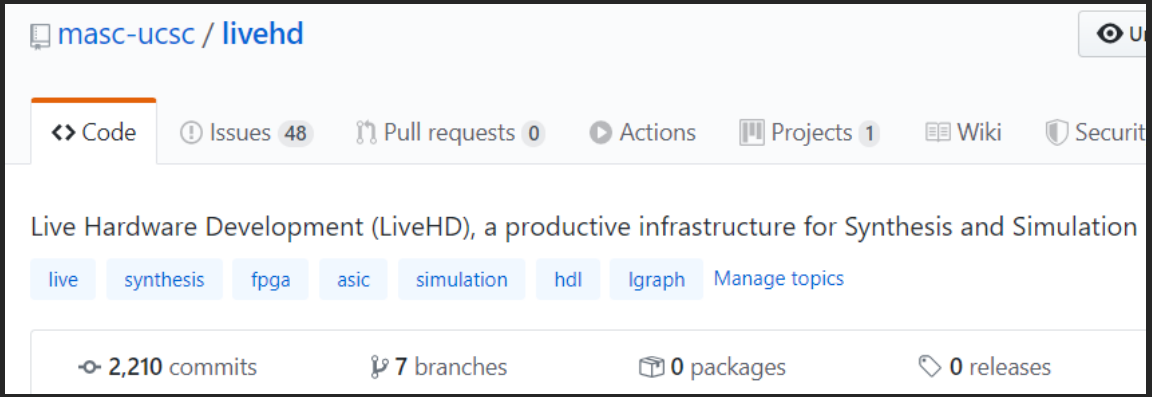


Pyrope can interact with other languages

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

Live Hardware Development (LiveHD)

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


masc-ucsc / livehd

Code | Issues 48 | Pull requests 0 | Actions | Projects 1 | Wiki | Securit

Live Hardware Development (LiveHD), a productive infrastructure for Synthesis and Simulation

live | synthesis | fpga | asic | simulation | hdl | lgraph | Manage topics

2,210 commits | 7 branches | 0 packages | 0 releases

- LGraph:
  - A graph/SSA netlist representation
  - Annotations, hierarchical traversal, topological sort…

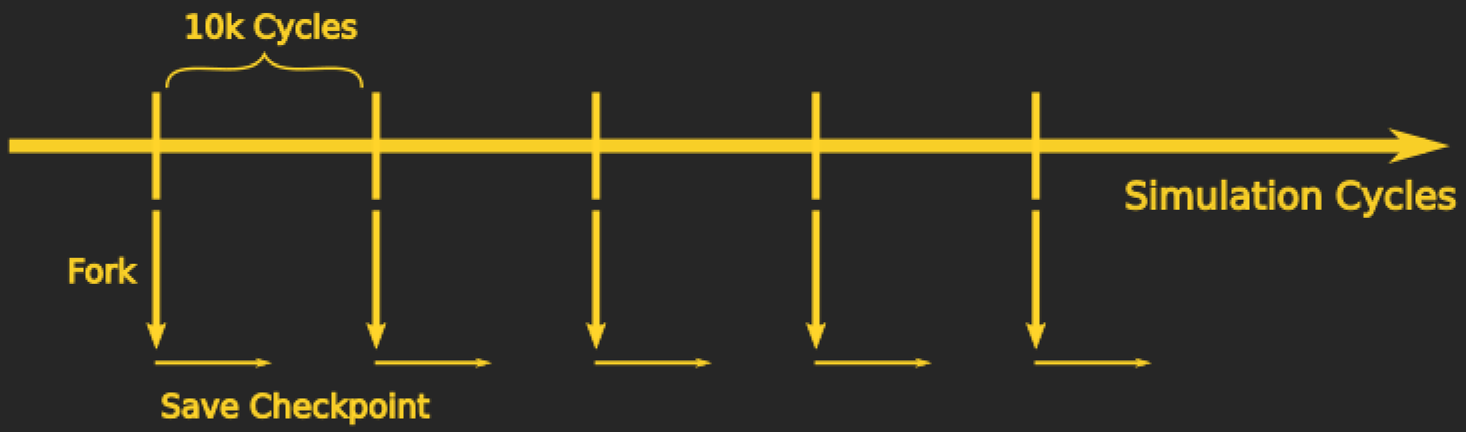
- LNAST: An neutral AST for hardware synthesis
  - Passes to/from Lgraph

Incremental flow

- Incremental/Live simulation with hot reload
- Incremental/Live synthesis FPGA place & route

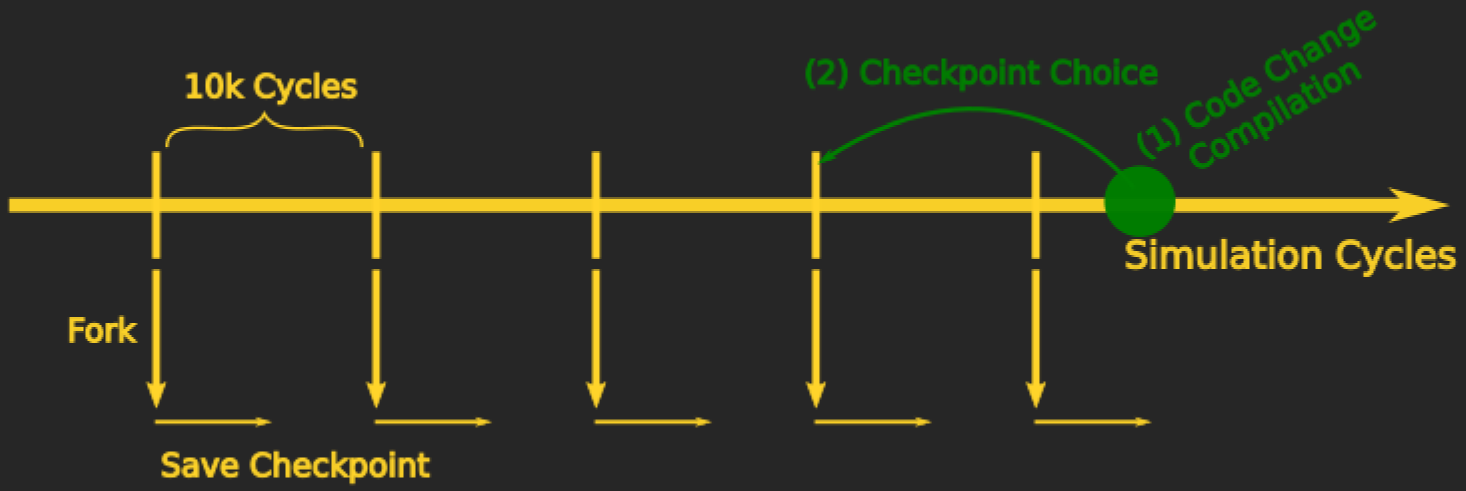
- We built prototypes in several papers
  - Now, we are working to bring them to LiveHD

Incremental simulation



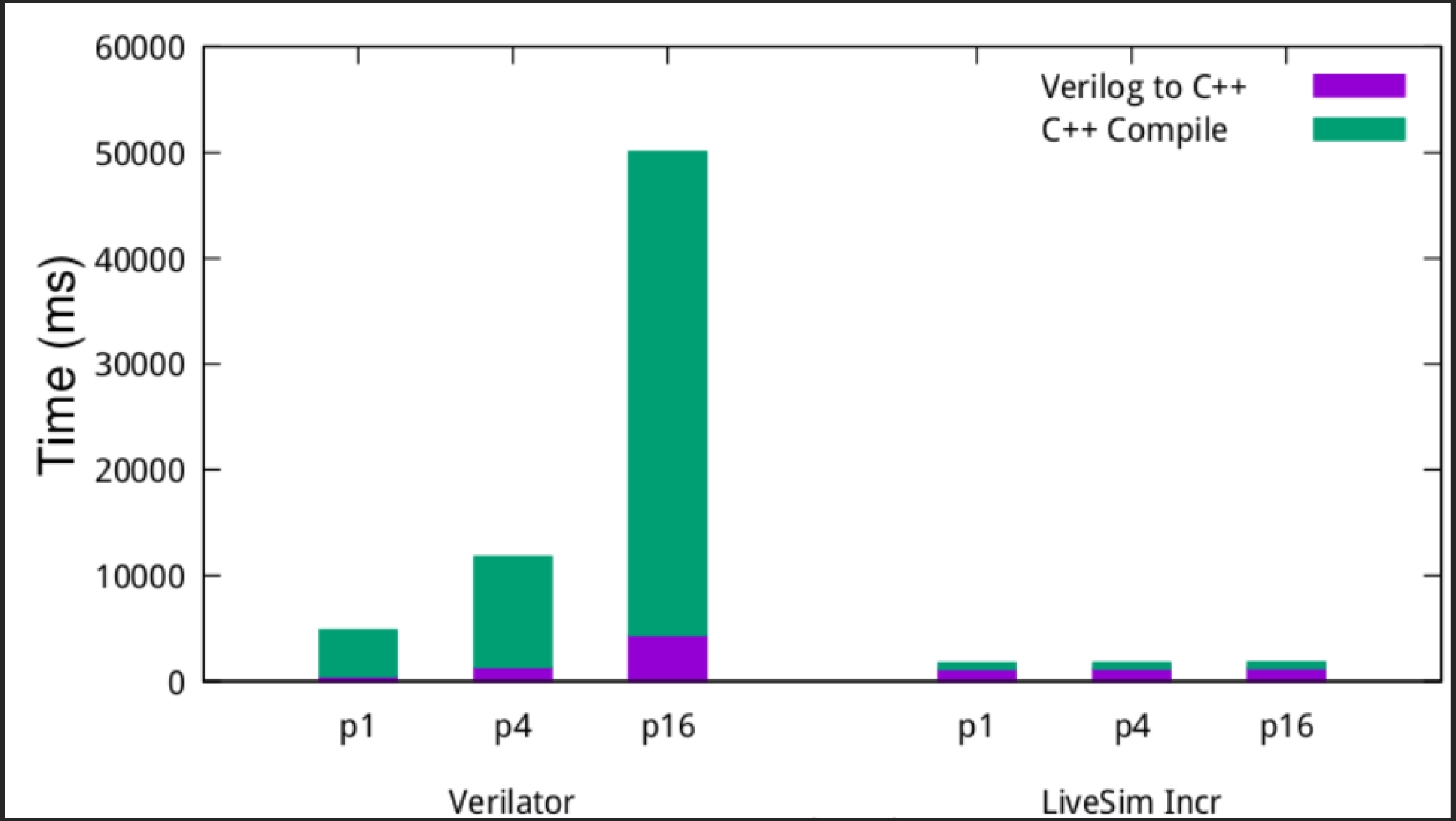
•LiveSim: A Fast Hot Reload Simulator, Haven Skinner, Rafael T. Possignolo, Sheng-Hong Wang, and Jose Renau, International Symposium on Performance Analysis of Systems and Software (**ISPASS**), April 2020.

Incremental simulation



• LiveSim: A Fast Hot Reload Simulator, Haven Skinner, Rafael T. Possignolo, Sheng-Hong Wang, and Jose Renau, International Symposium on Performance Analysis of Systems and Software (**ISPASS**), April 2020.

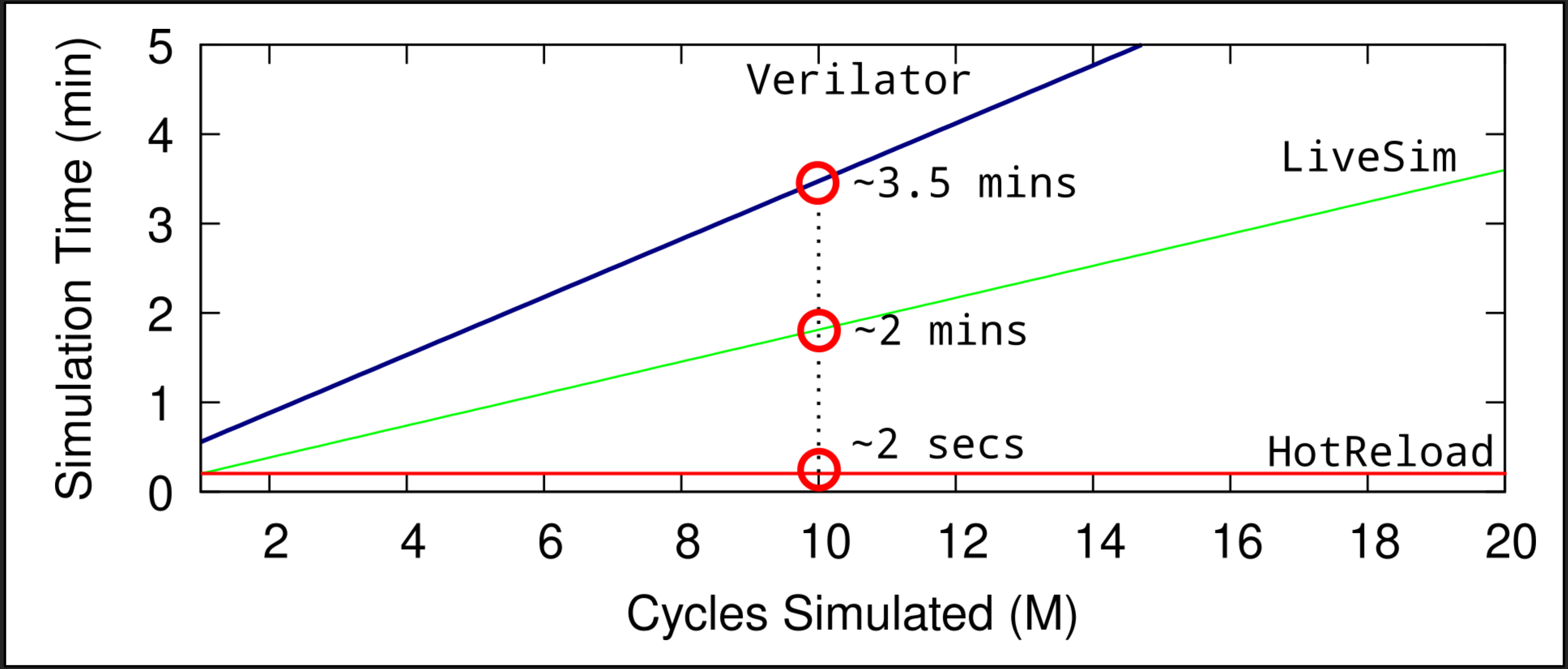
Incremental simulation generates C++



(1x1, 4x4, 16x16 core RISC-V SoC)

- LiveSim: A Fast Hot Reload Simulator, Haven Skinner, Rafael T. Possignolo, Sheng-Hong Wang, and Jose Renau, International Symposium on Performance Analysis of Systems and Software (**ISPASS**), April 2020.

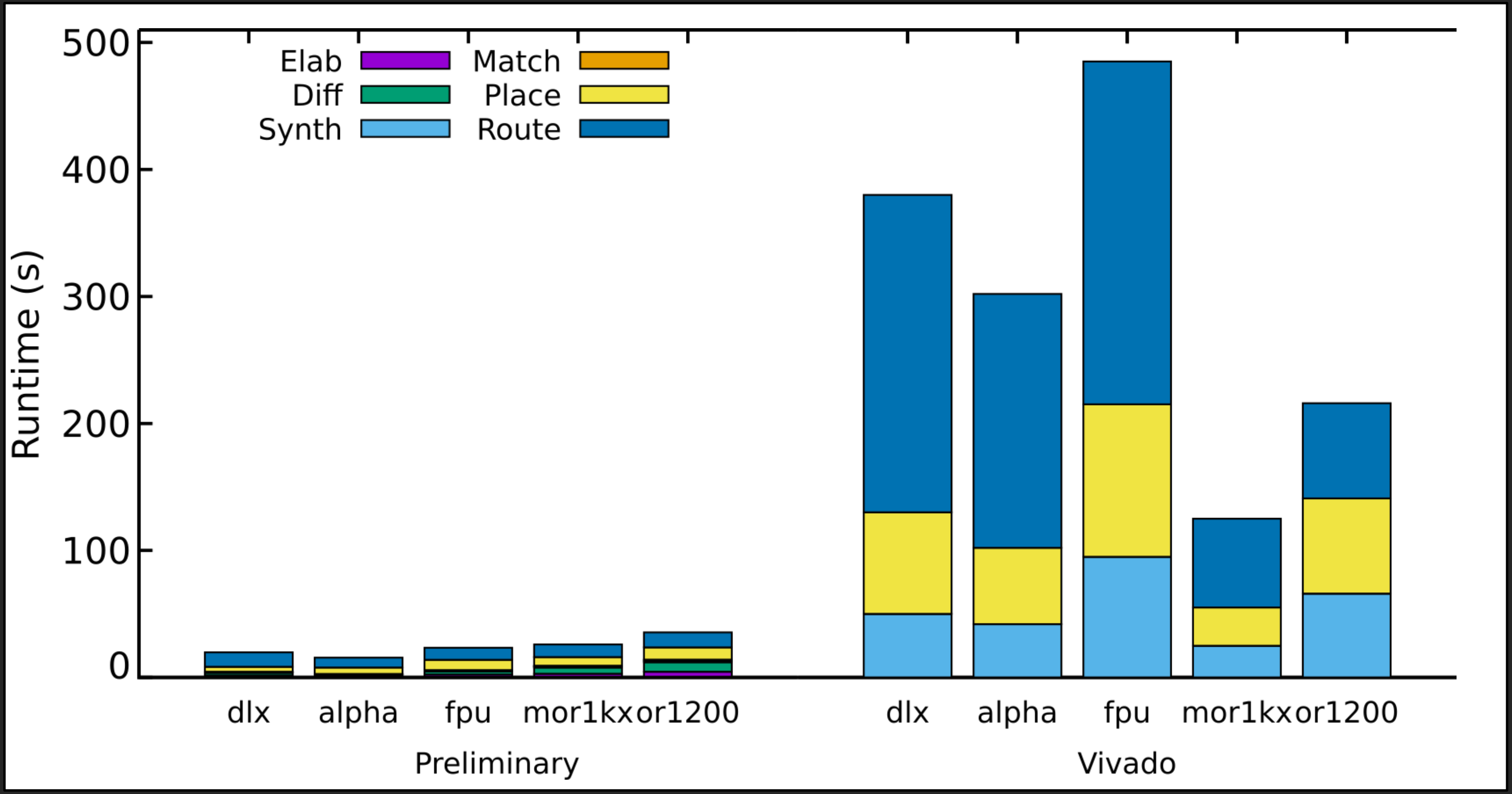
Incremental simulation with hot reload



(prototype still not released at github)

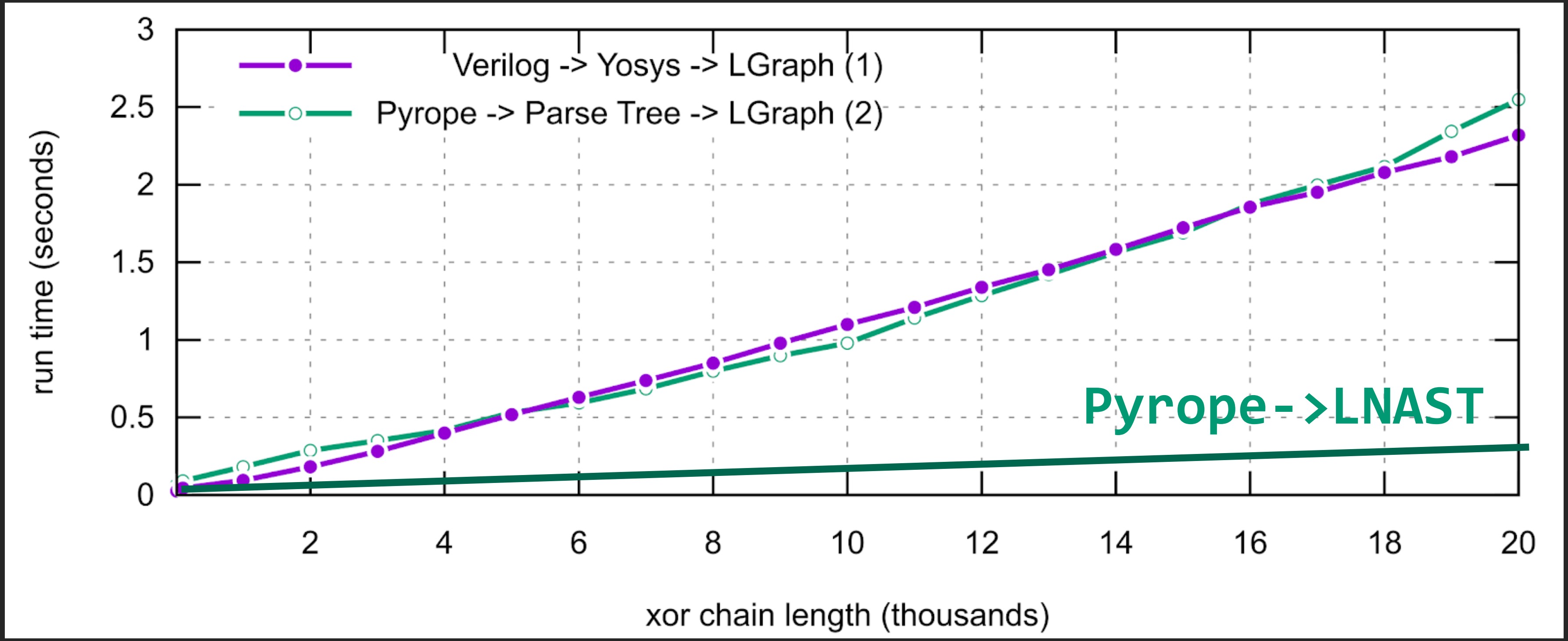
• LiveSim: A Fast Hot Reload Simulator, Haven Skinner, Rafael T. Possignolo, Sheng-Hong Wang, and Jose Renau, International Symposium on Performance Analysis of Systems and Software (**ISPASS**), April 2020.

Incremental synthesis and placement



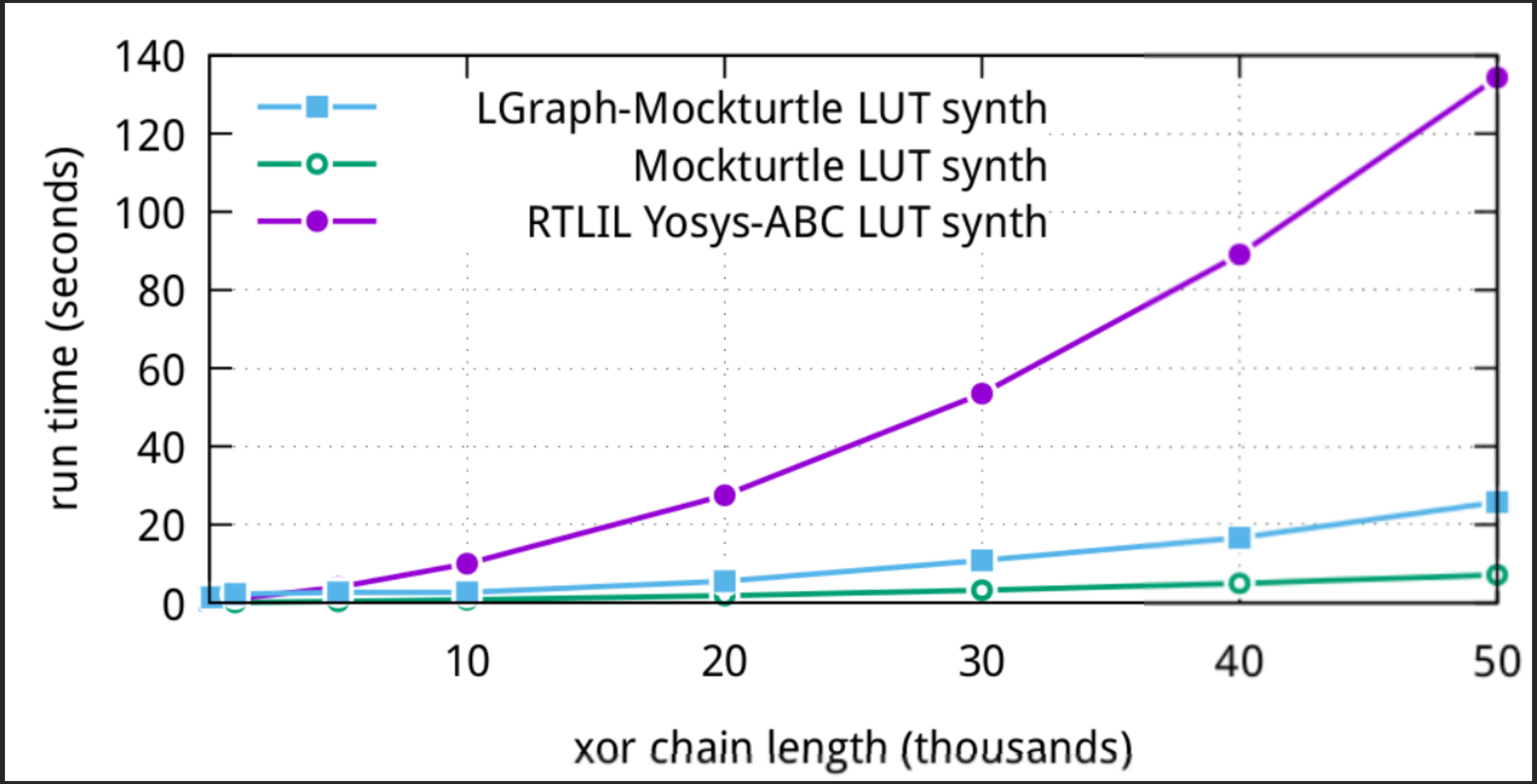
- SMatch: Structural Matching for Fast Resynthesis in FPGAs, Rafael T. Possignolo, and Jose Renau, Design Automation Conference (**DAC**), June 2019.

Scalable Verilog/Pyrope elaboration



- LGraph: A Unified Data Model and API for Productive Open-Source Hardware Design, Sheng-Hong Wang, Rafael T. Possignolo, Qian Chen, Rohan Ganpati, and Jose Renau, Second Workshop on Open-Source EDA Technology (**WOSET**), November 2019.
- LNAST: A Language Neutral Intermediate Representation for Hardware Description Languages, Sheng-Hong Wang, Akash Sridhar, and Jose Renau, Second Workshop on Open-Source EDA Technology (**WOSET**), 2019.

Scalable synthesis



- LGraph: A Unified Data Model and API for Productive Open-Source Hardware Design, Sheng-Hong Wang, Rafael T. Possignolo, Qian Chen, Rohan Ganpati, and Jose Renau, Second Workshop on Open-Source EDA Technology (**WOSET**), November 2019.
- LNAST: A Language Neutral Intermediate Representation for Hardware Description Languages, Sheng-Hong Wang, Akash Sridhar, and Jose Renau, Second Workshop on Open-Source EDA Technology (**WOSET**), 2019.

MASC Hardware Productivity Summary

- A productive hardware design flow
  - New HDL (Pyrope) integration
- Incremental
  - Elaboration
  - Synthesis
  - Simulation with Hot reload
- Scalable
  - Elaboration
  - Synthesis

LiveHD

Pyrope

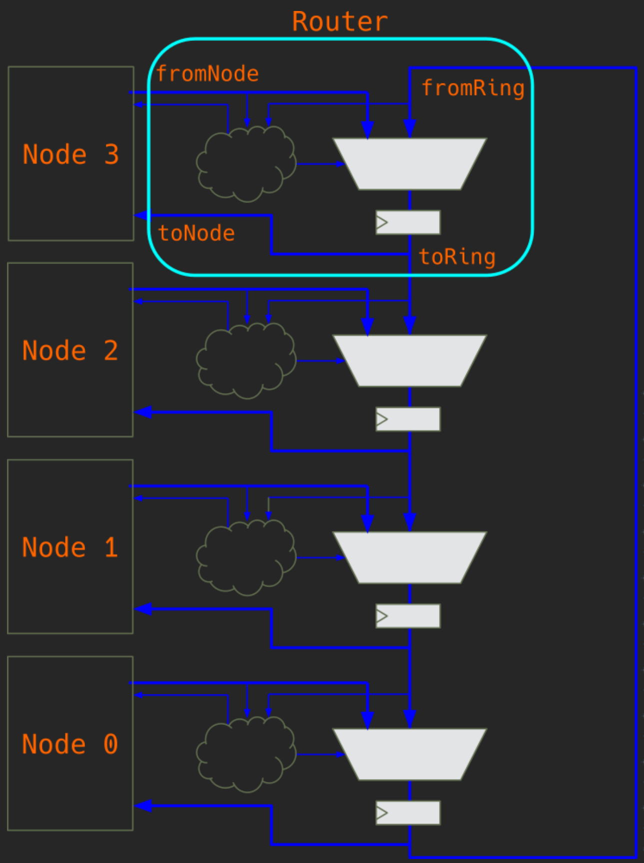
https://github.com/masc-ucsc/livehd

https://masc.soe.ucsc.edu/pyrope.html

- How could SAT solver for Logical Equivalence Checks help LiveHD/Pyrope?
- That way an incremental routing would be just as good as if it were done from a "make clean“
- difference between Fluid connections and pipelines as opposed to regular in comp arcitecture?
- What do "S", "P"&"R" represent in the FPGA flows

Backup Slides

Pyrope ring example



```
router = ::{
  if $fromRing? {                            // ? -> fluid pipeline
    if $fromRing.addr == $addr {
      %toNode     = $fromRing                // send to_node
      %toRing     = $fromNode                // new packet if present
    }else{
      %toRing     = $fromRing                // fwd packet
      $fromNode!  = true                     // back pressure from_node
    }
  }else{
    %toRing   = $fromNode                    // new packet if present
  }
  %.__fluid = true     // force fluid flops at the outputs
}

final = out.__final_value
for i in 0..3 {
  dst          := i + 1  // 0->1, 1->2, 2->3, 3->0
  packet.data = $from_node[i]
  packet.addr = i
  out[i]       = router(addr=i, toNode=final[dst].toNode, fromNode=packet)
  %to_node     = %to_node ++ out[i].toNode.data
}
```

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

Pyrope Integration with LiveHD example

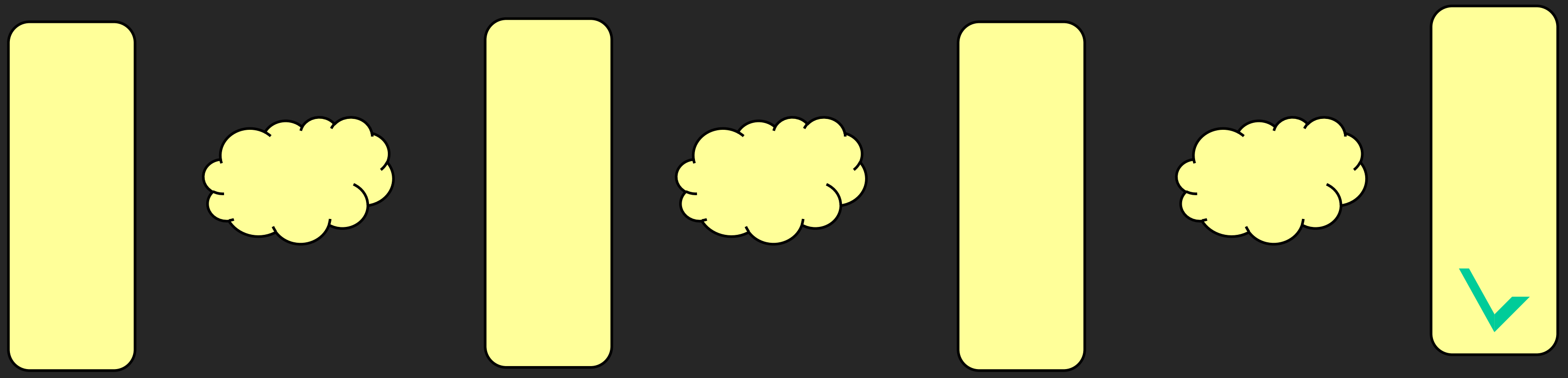
- Query modules in Pyrope

```
uarts = punch("*.uart_set")
offset = 0
for i in uarts {
  i.addr = 0x100 + offset
  offset += 16
}

assert(uarts.__size == 5)
```

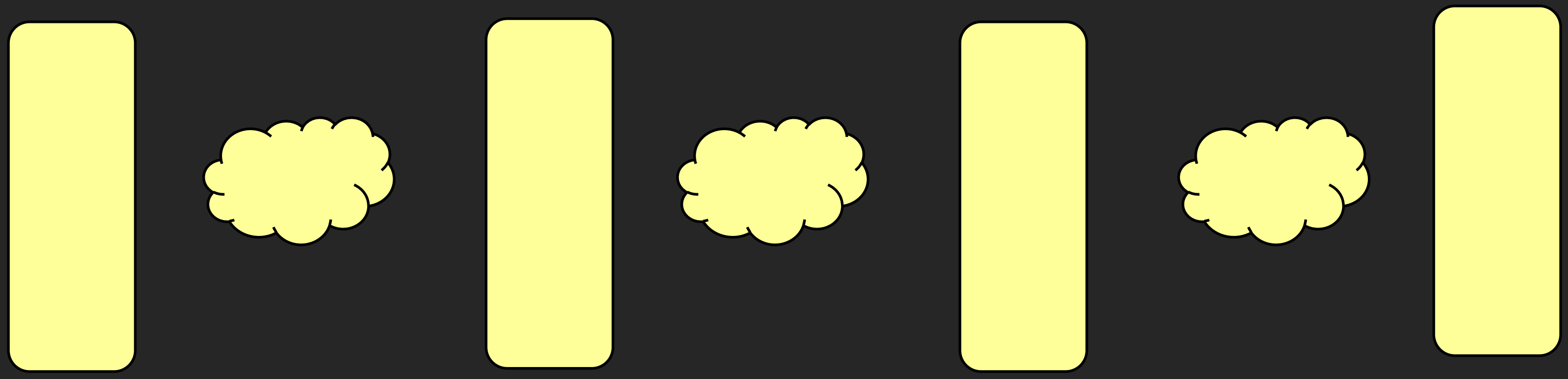
- Integrate with custom compiler passes

```
var.__poisoned = true


if foo.__poisoned {
}
```

Simple straight example

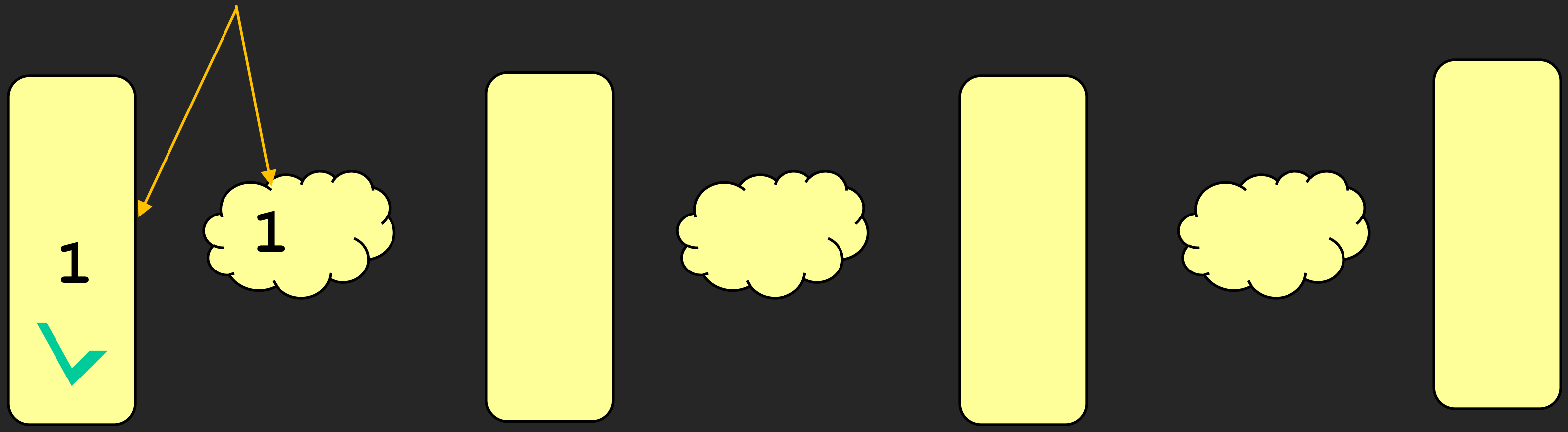


Simple straight example

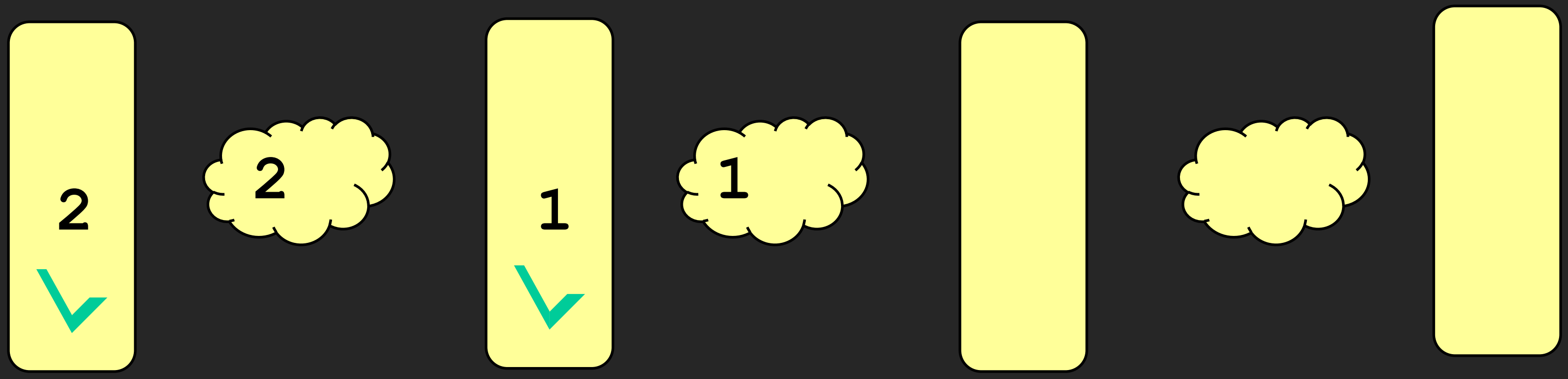


Simple straight example

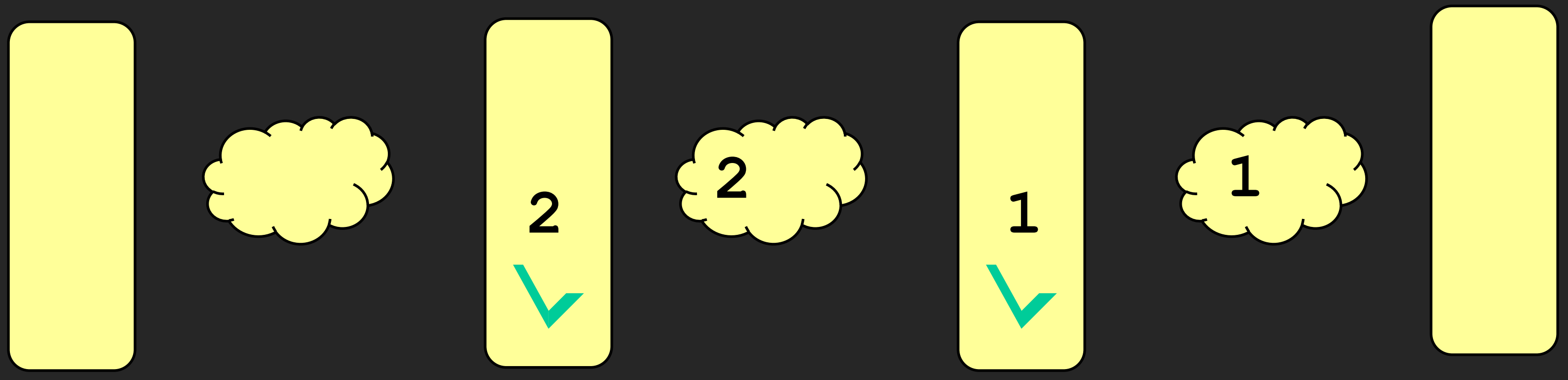
“1” is the value visible at the output of the fluid flop, and hence the combinational logic in the stage



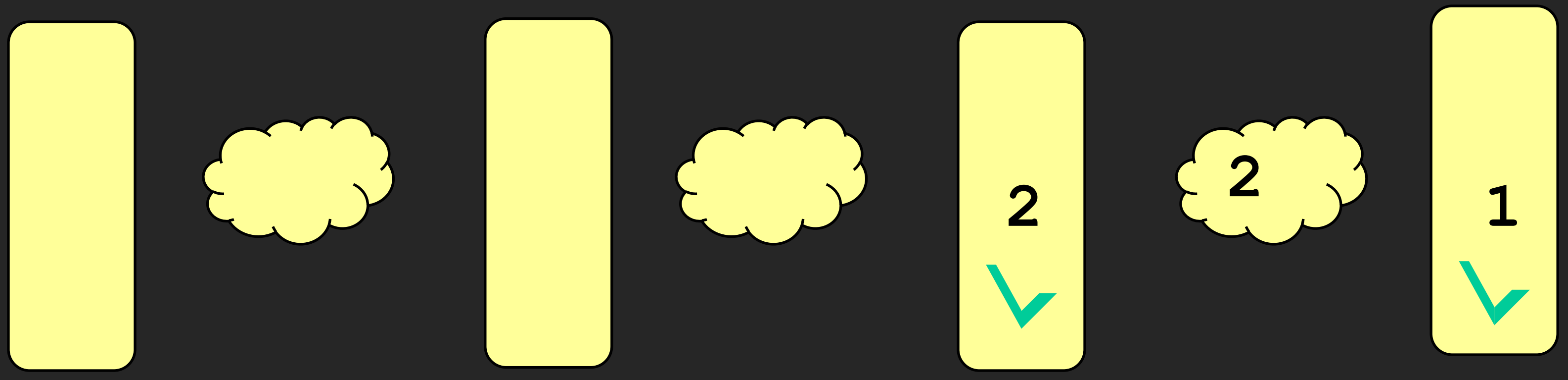
Simple straight example



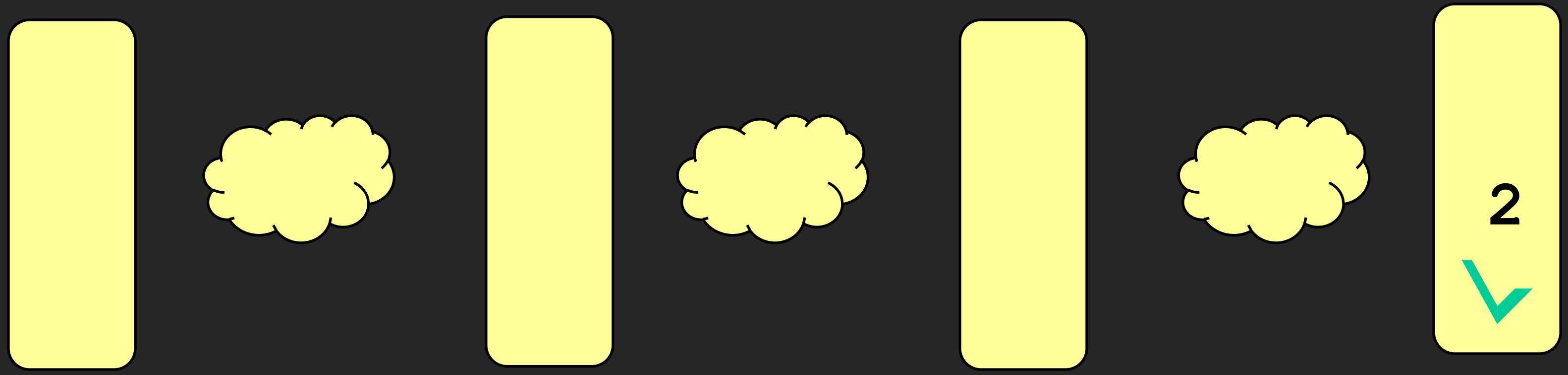
Simple straight example



Simple straight example

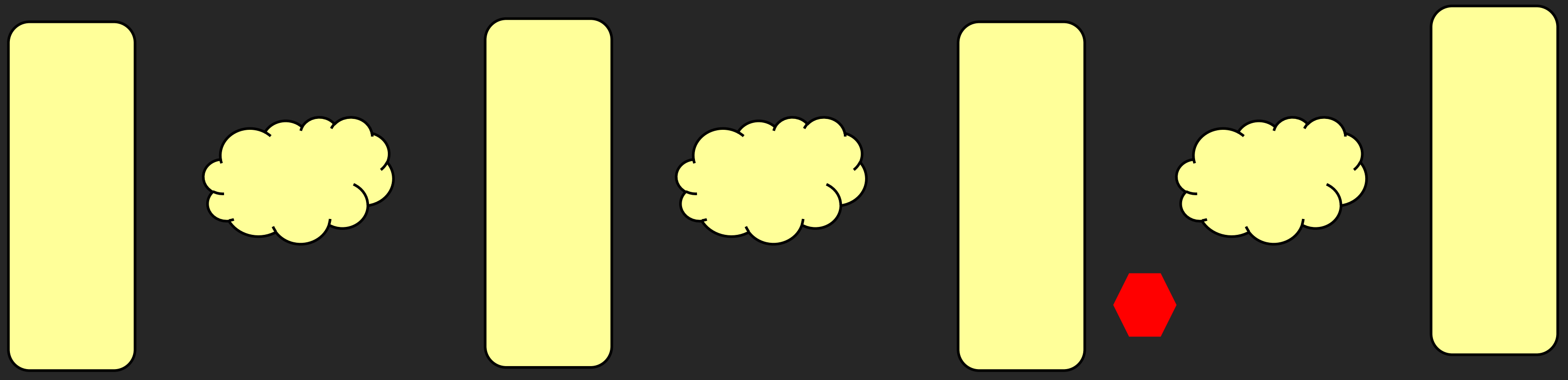


Simple straight example



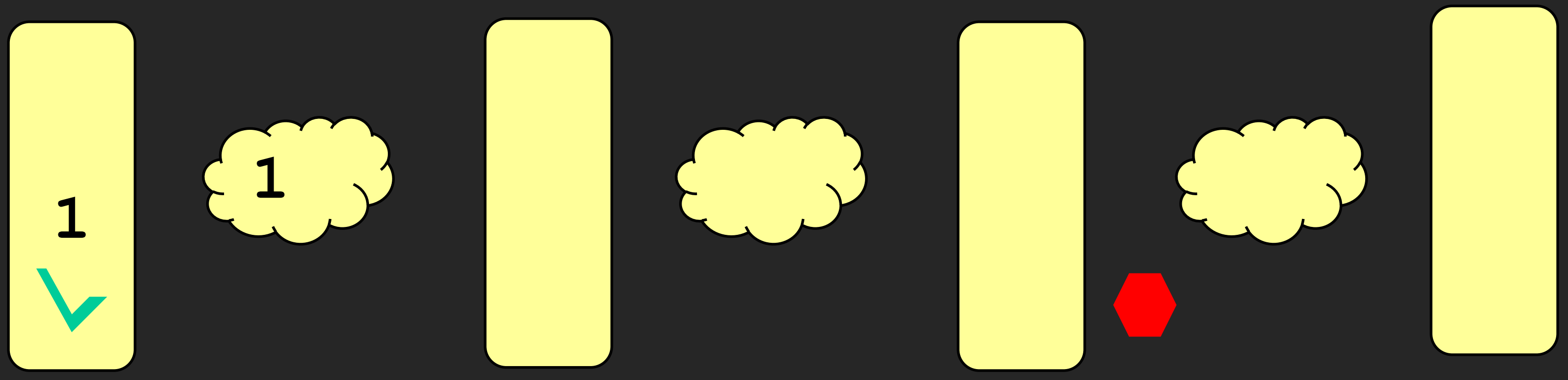
Simple straight example

Simple straight example

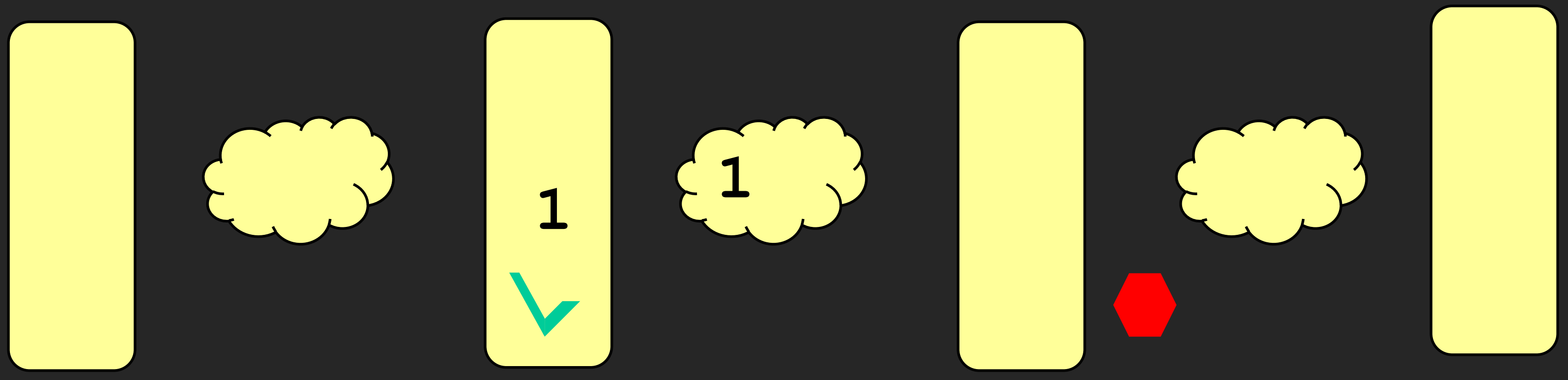


Retry/STOP asserted for some reason

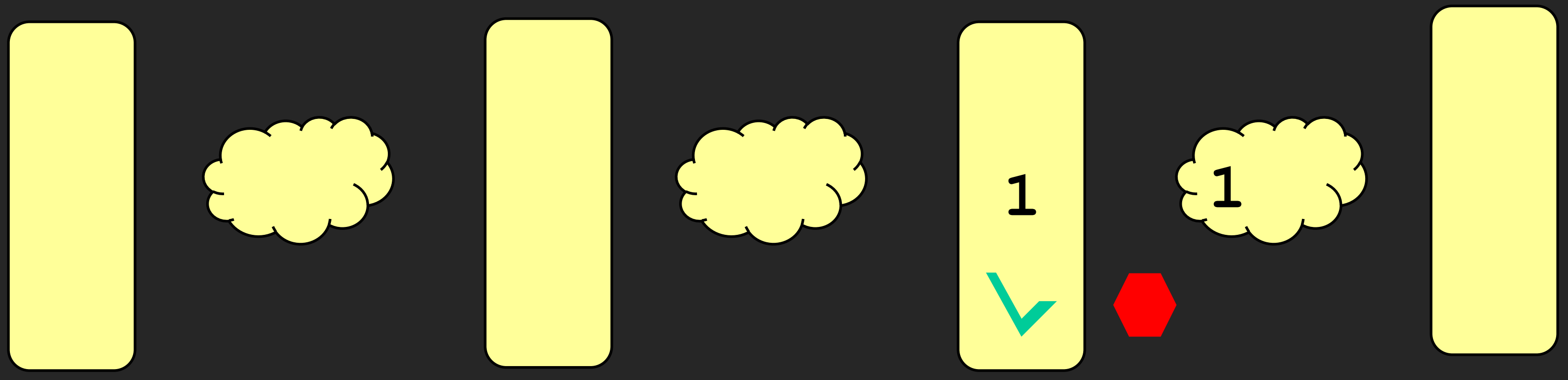
Simple straight example



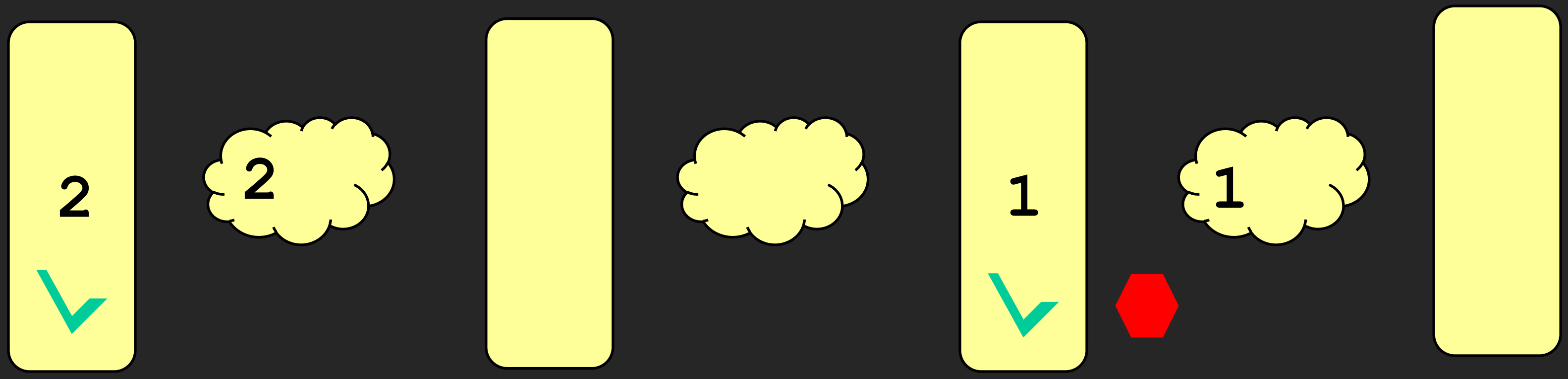
Simple Straight Example



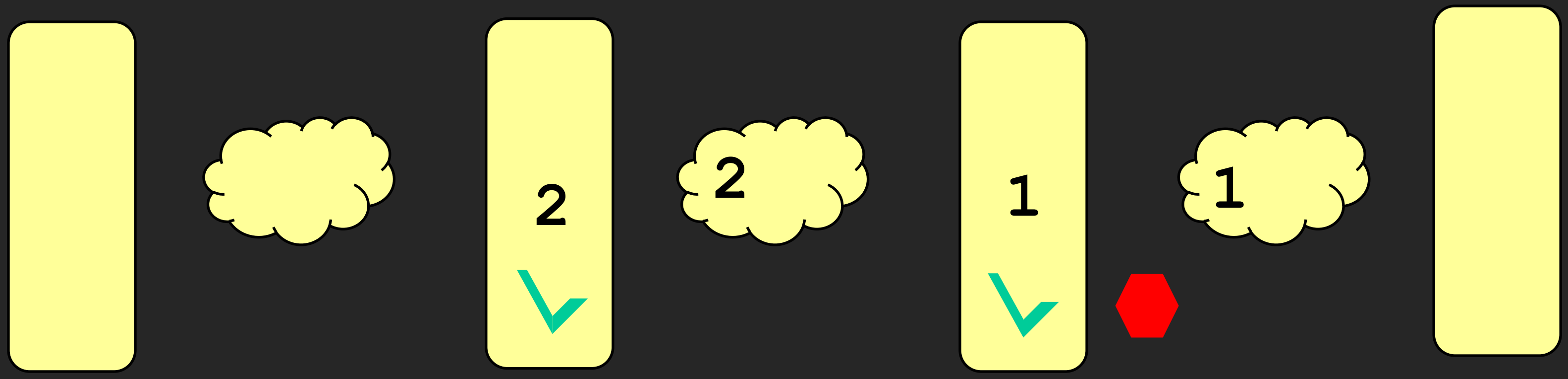
Simple straight example



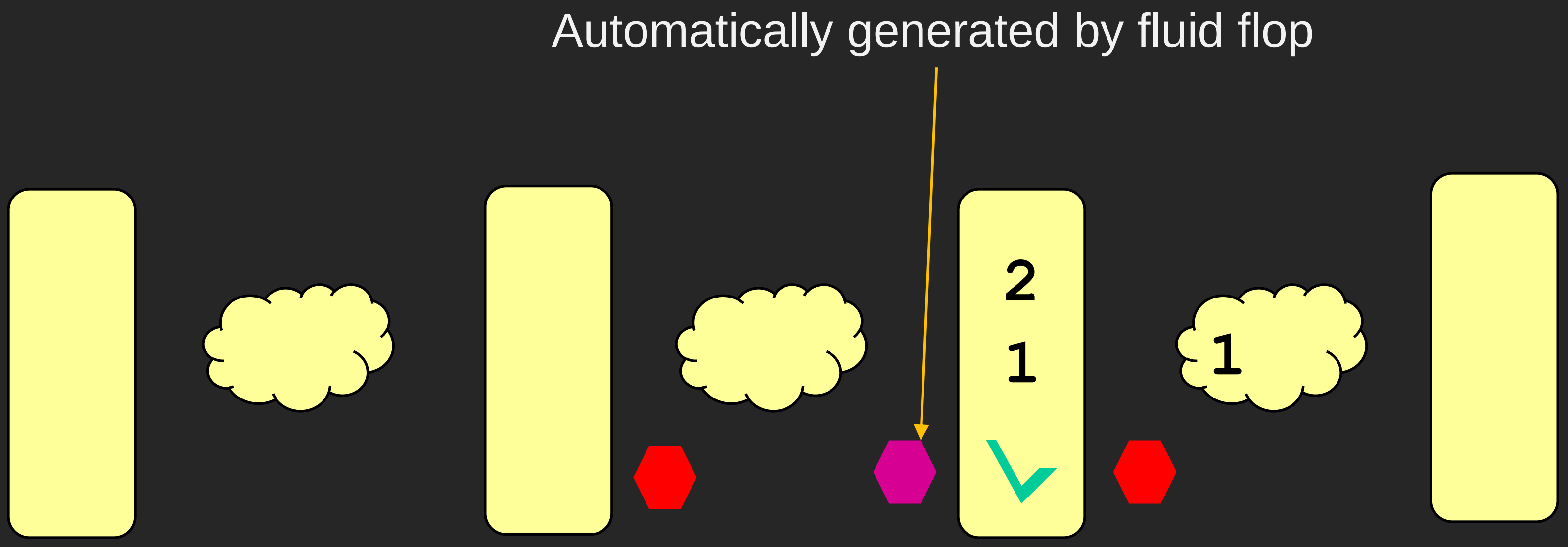
Simple straight example



Simple straight example

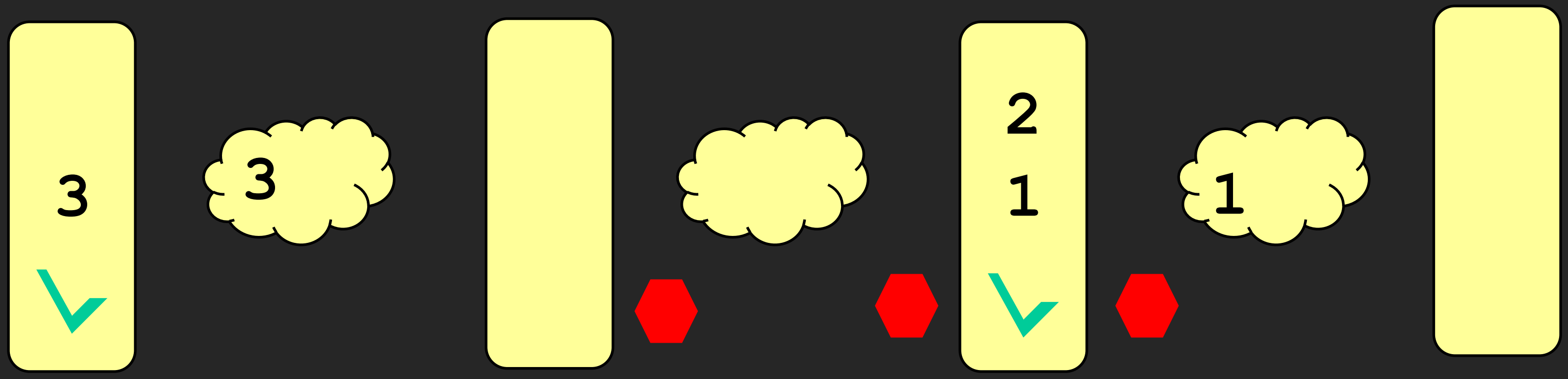


Simple straight example

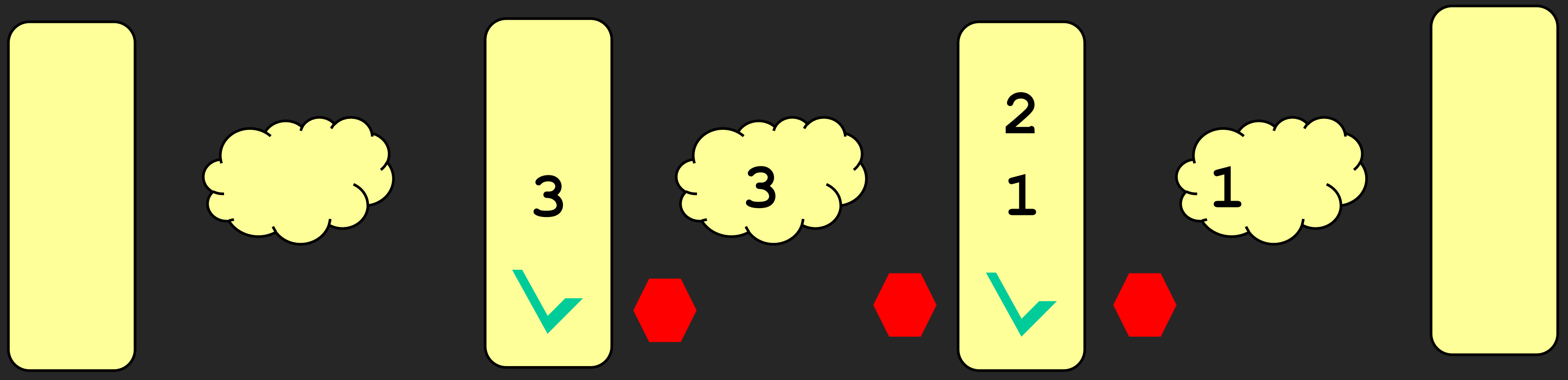


1 is stalled in the fluid flop, and 2 is kept in the shadow copy. Therefore, the “fifo” is full and stop is propagated to the previous stage

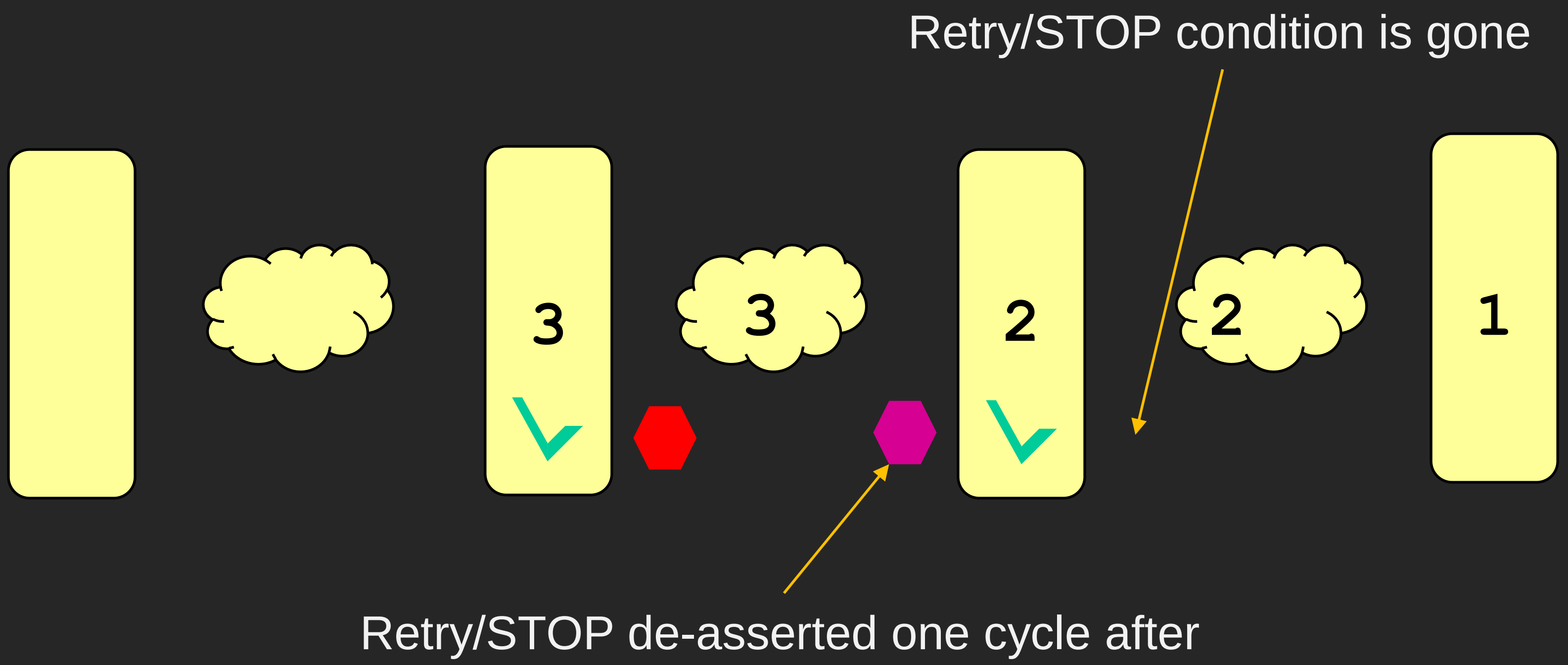
Simple straight example



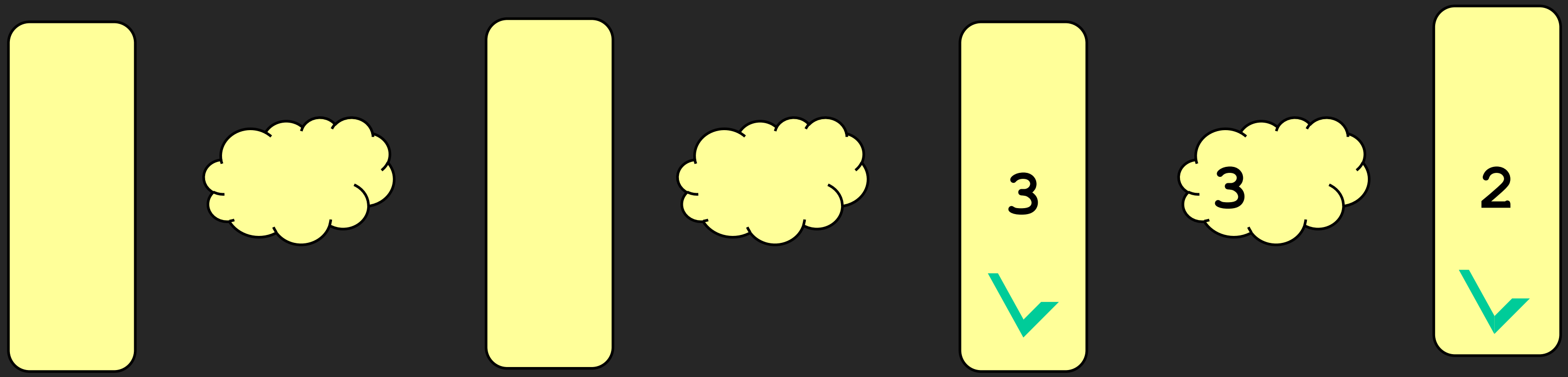
Simple straight example



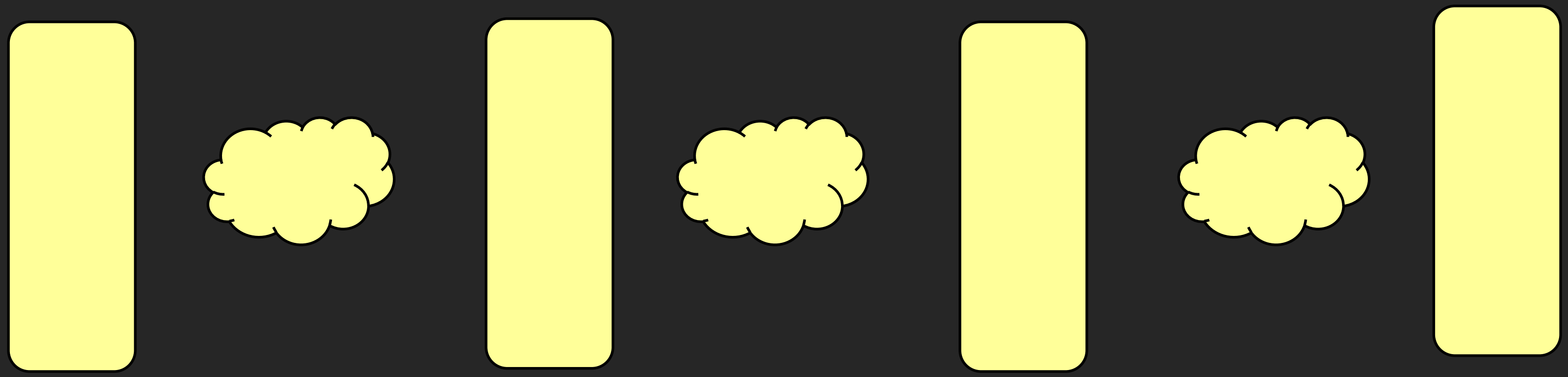
Simple straight example


Retry/STOP condition is gone
3
3
2
2
1
Retry/STOP de-asserted one cycle after

Simple straight example



Simple straight example

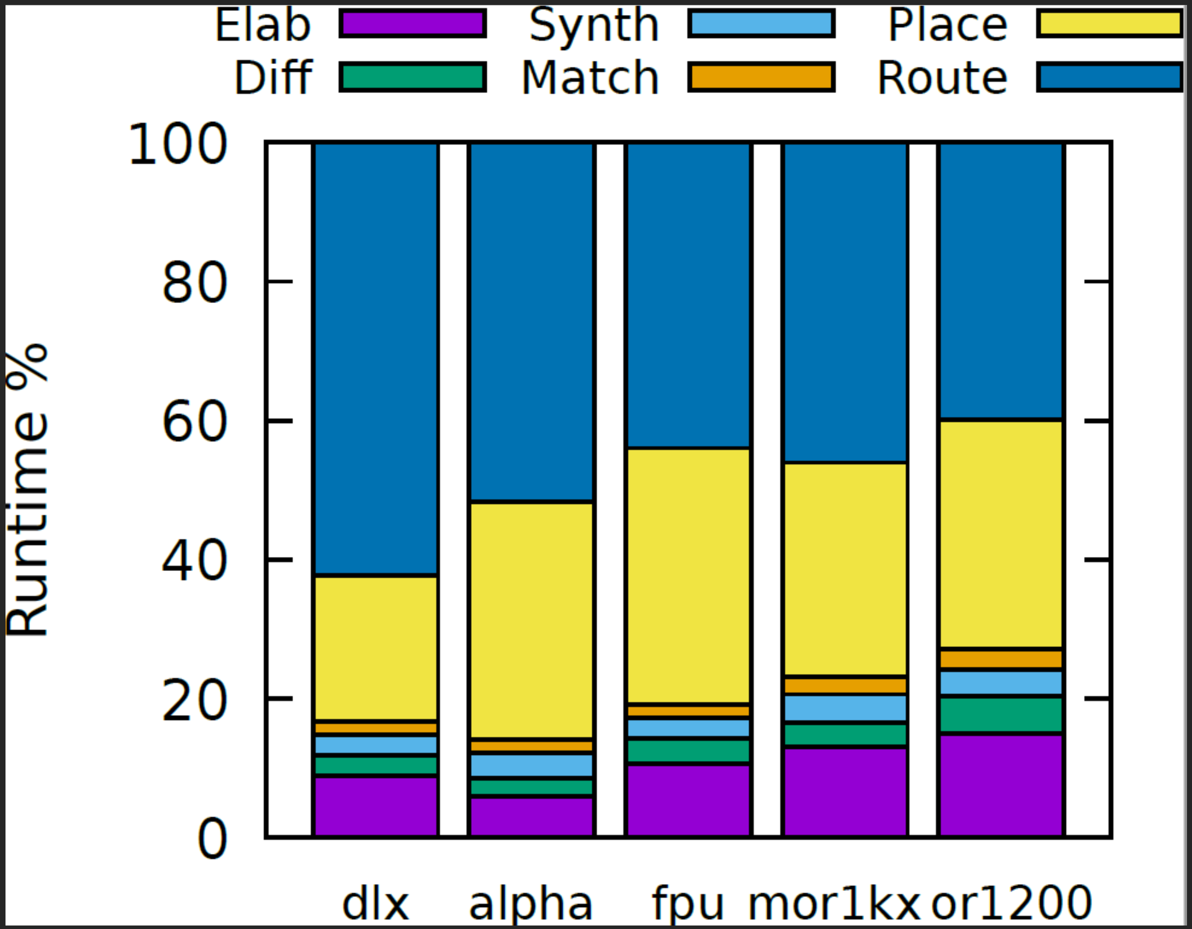


Pyrope Alpha (2008-2018)

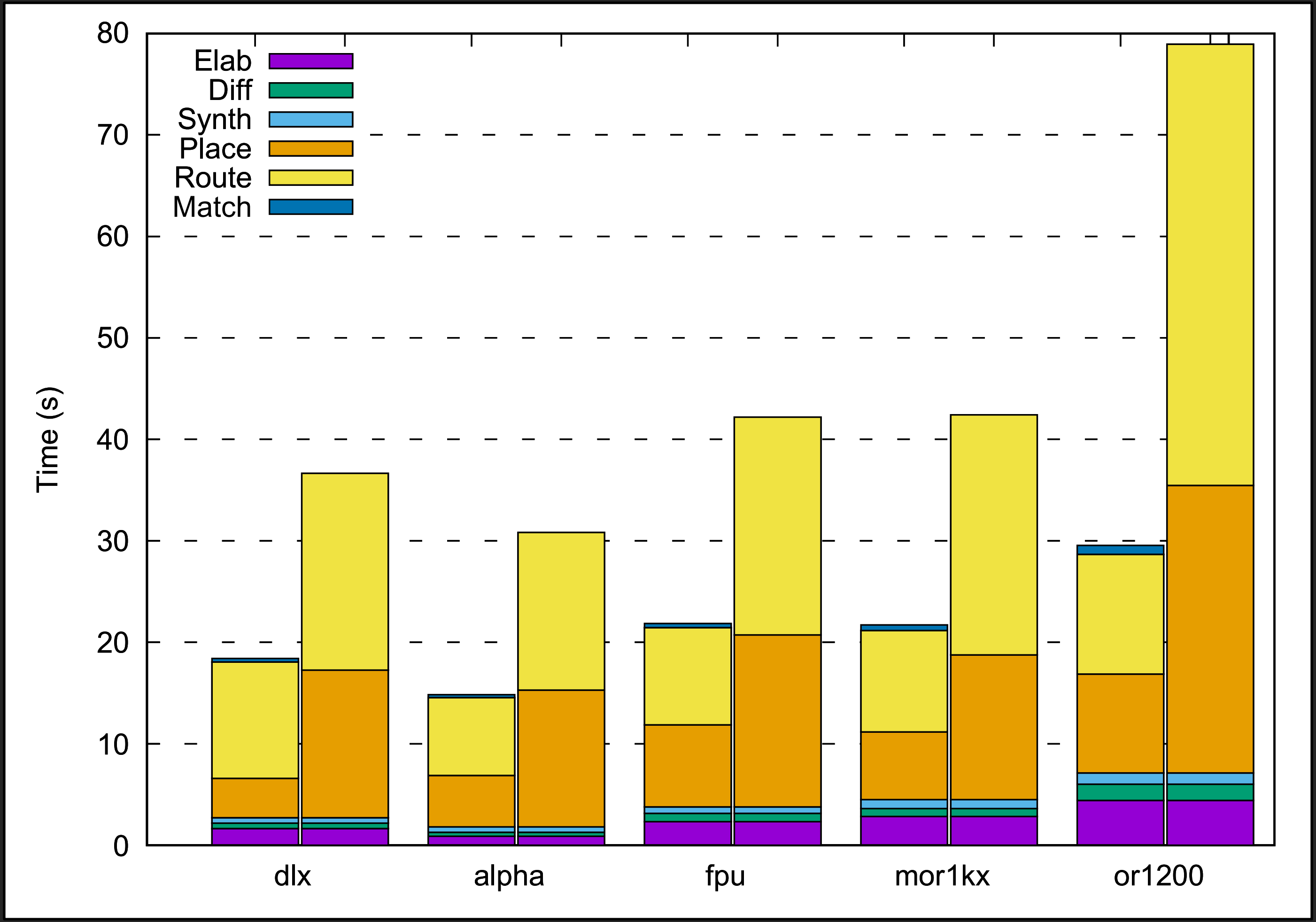
- Several designs implemented
  - RISC-V core, mesh, Fpunit…

```
if $inst?:
  opcode = $inst{INST_OPCODE}

  if (opcode == ARITH_REG_OP) or (opcode == ARITH_REG_W_OP):
    ra1 = $inst{INST_OP1}
    ra2 = $inst{INST_OP2}

    if @raw{ra1} == 1 or @raw{ra2} == 1: // check for Read-After-Write
      keep $inst
```

Time breakdown FPGA incremental



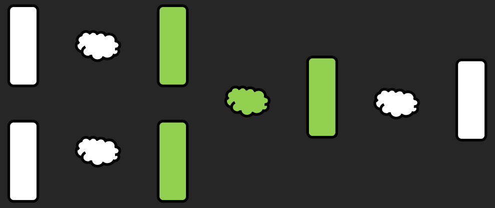
Total synth+place+route for FPGAs



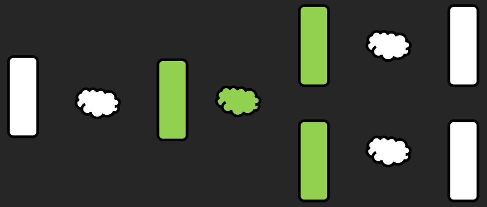
The match overhead is small

Fluid connections

- Straight:
  - One Fluid Flop connects to 1 Fluid Flop (1:1)
- Join:
  - Two Fluid Flops used to generate a value in one flow (2:1)

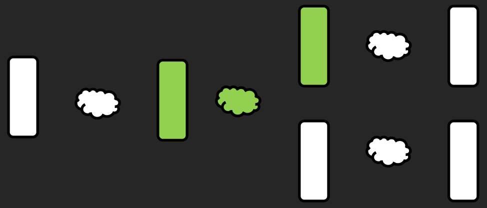


- Fork:
  - One Fluid Flop goes to 2 output flops (2:1)

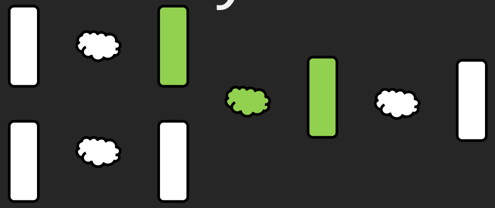


Fluid connections II

- Branch:
  - Propagate data to only one of the output fluid flops



- Merge:
  - Pick data from only one of the input fluid flops



Fluid in an HDL

- Common handshake simplifies “concepts”
- Tool can leverage handshake to optimize
- Verilog can be “error prone” to code this

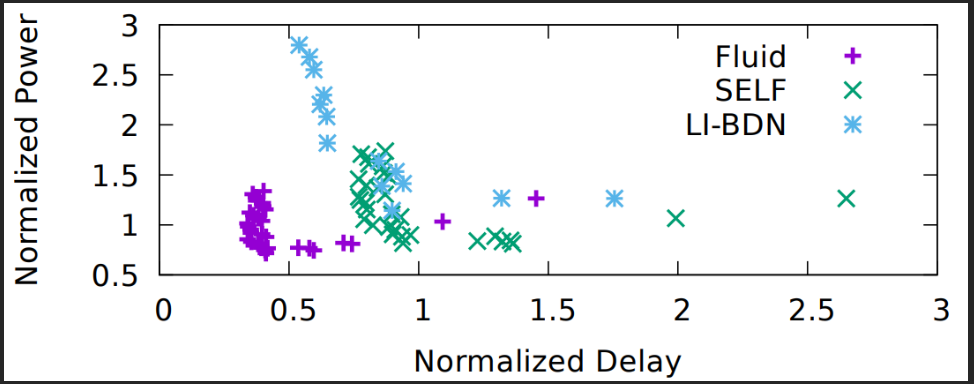
- Pyrope Beta Fluid Support
  - Actor model for fluid

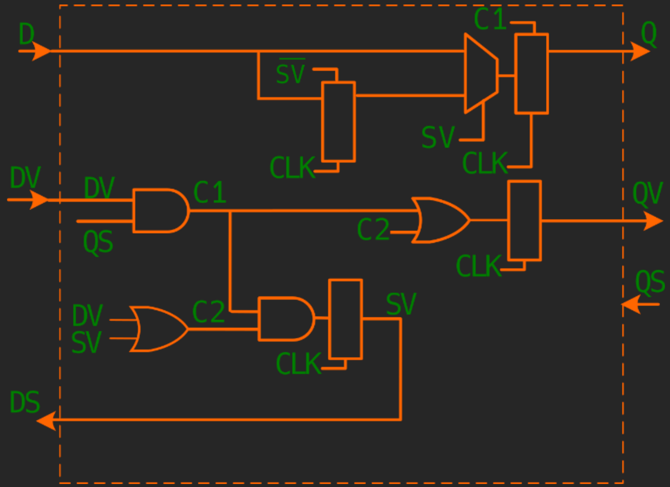
```
if a? and a.counter>0 { // Option 1
   #total += a.counter
}
try {                   // Option 2 (same)
  if a.counter>0 {
    #total += a.counter
  }
}
if a?.counter>0 {      // Option 3 (same)
  #total += a.counter
}
#total += a?.counter   // Option 4 (same)
```

Fluid Pipelines

- Fluid Pipeline is a handshake between pipeline stages
  - https://github.com/masc-ucsc/fluid

Automated Pipeline Transformations with Fluid Pipelines, Rafael T. Possignolo, Elnaz Ebrahimi, Haven Skinner, Jose Renau. Advanced Logic Synthesis (**Book Chapter**), December 2017.
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.


Normalized Power
Normalized Delay
Fluid
SELF
LI-BDN


D
Q
C1
$\overline{SV}$
SV
CLK
DV
QV
QS
C2
DS