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
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

https://github.com/masc-ucsc/lgraph
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
# Some Open Source Tools for HW

<table>
<thead>
<tr>
<th>HDL</th>
<th>Front End</th>
<th>Back End</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyMTL</td>
<td>Verilator</td>
<td>Ophidian</td>
</tr>
<tr>
<td>CHISEL</td>
<td>Yosys</td>
<td>Rsyn</td>
</tr>
<tr>
<td>Pyrope</td>
<td>ABC</td>
<td>Arachne-pnr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Next-pnr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VTR</td>
</tr>
</tbody>
</table>

https://github.com/masc-ucsc/lgraph
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](https://github.com/RsynTeam/rsyn-x)
• FIRRTL, [https://github.com/freechipsproject/firrtl](https://github.com/freechipsproject/firrtl)
• Yosys, [https://github.com/YosysHQ/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++…)

```
LGraph
Pass
(transformation)
```

- Synthesis
  - Yosys
  - ABC
  - Rapid
  - Wright
  - ...

- Compiler
  - DCE
  - Bitwidth
  - Pyrope
  - Compiler
  - ...

- HDL
  - Verilog
  - Pyrope
  - *CHISEL*

- Design Library
  - LEF/DEF
  - BLIF
  - Liberty
  - ...

- Code Gen.
  - Verilog
  - Pyrope
  - C++
UCSC Components on Live Hardware

• LGraph [https://github.com/masc-ucsc/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](https://masc.soe.ucsc.edu/pyrope.html)
Some Results
LGraph is Fast!

![Bar chart showing time in seconds for different operations (Execute, Write, Read) for various datasets (sb1, sb3, sb4, sb5, sb7, sb10, sb16, sb18).

- **LGraph**:
  - sb1: 264s
  - sb3: 262s
  - sb4: 167s
  - sb5: 208s
  - sb7: 65s
  - sb10: 464s
  - sb16: 63s
  - sb18: 533s

- **RSyn**:
  - sb1: 189s
  - sb3: 147s
  - sb4: 189s
  - sb5: 147s
  - sb7: 189s
  - sb10: 147s
  - sb16: 189s
  - sb18: 147s

- **Yosys**:
  - sb1: 189s
  - sb3: 147s
  - sb4: 189s
  - sb5: 147s
  - sb7: 189s
  - sb10: 147s
  - sb16: 189s
  - sb18: 147s

- **Time in seconds**
  - Y-axis: 0, 10, 20, 30, 40, 50

- **Legend**:
  - Execute: purple
  - Write: green
  - Read: orange

- **Dataset**: sb1, sb3, sb4, sb5, sb7, sb10, sb16, sb18

---

Jose Renau

https://github.com/masc-ucsc/lgraph
ASIC (DC) Incremental Synthesis under 60 secs

Baseline

29 minutes

Worst case

5 minutes

Typical

1 minute
Live FPGA
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

Github LoC per month

<table>
<thead>
<tr>
<th>Tool</th>
<th>LoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ophidian</td>
<td>9K C++</td>
</tr>
<tr>
<td>TaskFlow</td>
<td>13K C++</td>
</tr>
<tr>
<td>OpenTimer</td>
<td>15K C++</td>
</tr>
<tr>
<td>Qrouter</td>
<td>18K C</td>
</tr>
<tr>
<td>RapidWright (Xilinx)</td>
<td>24K Java</td>
</tr>
<tr>
<td>Netgen (LVDS)</td>
<td>32K C</td>
</tr>
<tr>
<td>LGraph</td>
<td>36K C++17</td>
</tr>
<tr>
<td>Rsyn-x (including GUI)</td>
<td>52K C++</td>
</tr>
<tr>
<td>Yosys</td>
<td>78K C++</td>
</tr>
<tr>
<td>Graywolf (placer)</td>
<td>82K C</td>
</tr>
<tr>
<td>ABC (no vudd, zlib…)</td>
<td>488K C</td>
</tr>
</tbody>
</table>
Graphviz Output Pass Example

- 105 LoC
- 23 Header
- 82 Code
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…

- 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
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

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.

[Diagram showing the Live Synthesis Flow]

- Setup Phase
  - Initial Synthesis
  - Elaboration
  - Synthesis

- Setup Pass
  - Functional Match
  - Place & Route

- Live Phase
  - Interactive
    - ΔHDL
    - ΔElaboration
    - Netlist Diff
    - ΔSynthesis
    - Netlist Stitch
  - Place & Route
How big are the changes?

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

<table>
<thead>
<tr>
<th>Cone Size</th>
<th>fpu</th>
<th>mips</th>
<th>or1200</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;200</td>
<td>1769</td>
<td>1237</td>
<td>643</td>
</tr>
<tr>
<td>200-300</td>
<td>99</td>
<td>73</td>
<td>172</td>
</tr>
<tr>
<td>300-400</td>
<td>938</td>
<td>35</td>
<td>156</td>
</tr>
<tr>
<td>400-500</td>
<td>1</td>
<td>2</td>
<td>185</td>
</tr>
<tr>
<td>500-600</td>
<td>649</td>
<td>11</td>
<td>74</td>
</tr>
<tr>
<td>600-800</td>
<td>34</td>
<td>316</td>
<td>63</td>
</tr>
<tr>
<td>800-1000</td>
<td>33</td>
<td>29</td>
<td>58</td>
</tr>
<tr>
<td>1000-1500</td>
<td>5</td>
<td>124</td>
<td>56</td>
</tr>
<tr>
<td>1500-2000</td>
<td>1</td>
<td>550</td>
<td>0</td>
</tr>
<tr>
<td>2000-3000</td>
<td>0</td>
<td>421</td>
<td>0</td>
</tr>
<tr>
<td>3000-4000</td>
<td>0</td>
<td>302</td>
<td>0</td>
</tr>
<tr>
<td>&gt;4000</td>
<td>0</td>
<td>115</td>
<td>0</td>
</tr>
</tbody>
</table>
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](https://github.com/masc-ucsc/fluid)
- A new elastic pipeline methodology to have composable transformations

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](image1)

![Latch based implementation](image2)
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](https://masc.soe.ucsc.edu/pyrope.html)