Industry-Grade SystemVerilog IPs And The Open Flow: How We Synthesized Iguana

Integrated Systems Laboratory (ETH Zürich)

Thomas Benz  tbenz@iis.ee.ethz.ch
Paul Scheffler  paulsc@iis.ee.ethz.ch
Jannis Schönleber  janniss@iis.ee.ethz.ch
Philippe Sauter  phsauter@ethz.ch

FSiC2023, Paris, July 2023

PULP Platform
Open Source Hardware, the way it should be!
Our Goal

- **An end-to-end open Linux-capable ASIC (IHP 130nm)**
  - Open hardware (RTL), software/OS, EDA tools, PDK, standard cells

- **During the course of this project: increase openness**
Current Support In Open Synthesis

How well will Yosys handle complex designs?

- **RTL**
  - Manage
  - Process

- **Synthesis**
  - Parse
  - Optimize
  - Elaborate
  - Tech Map

- **Backend**
  - Floorplan
  - Clock Tree
  - Placement
  - Routing

- **OpenROAD**

- **Geometry**
  - DRC/LVS
  - Format

---

**yosys – Yosys Open SYnthesis Suite**

This is a framework for RTL synthesis tools, it currently has extensive Verilog-2005 support and provides a rich set of synthesis capabilities.

- Natively: very solid Verilog support. (But only limited SV)
- There are solutions out there! RTL pre-processing

---

FSiC2023, Paris, July 2023
Parallel Ultra Low Power (PULP) Platform

• Research on open-source energy-efficient computing architectures

• Started in 2013, we are celebrating 10 years of our project this year

• Led by Luca Benini

• Involves ETH Zürich (Switzerland) and University of Bologna (Italy)

• Large group of almost 100 people

FSiC2023, Paris, July 2023
We Have Designed And Tested More Than 50 PULP ICs

Check http://asic.ethz.ch for all our chips
All Of Our Designs Are Open Source Hardware

- All our development is on GitHub using a permissive license
  - HDL source code, testbenches, software development kit, virtual platform

https://github.com/pulp-platform

- Allows anyone to use, change, and make products without restrictions.
SystemVerilog, Why Bother?

- There are alternatives
  - Chisel, SpinalHDL, Amaranth

- We use it, a lot
  - The top language in our repos
  - 1M+ lines of code
  - All our systems in SV
  - Silicon-proven in 50+ PULP tapeouts
  - Logical choice for Iguana/Tegu/Komodo

- Widespread in industry
  - Many commercial IPs in SV

- Works in proprietary tools
Iguana: A Linux-Capable Design

- **We choose a complex SoC design**
  - Academy & industry: complexity **increases**
  - More **compute** & **complex interfaces**

- **Evaluating the open flow requires complexity**
  - **Sufficient** hardware **complexity**
    - Full Linux-capable **CVA6** SoC: ~2 MGE
    - Arithmetic operands, std-cell memory, ROMs
    - A **capable** demonstrator chip
  - **Multiple** clock domains (HyperBus, C2C link)
    - Implementation & constraining of CDCs
  - **High-speed**, high-complexity async interfaces
    - Synthesis of DDR/asynchronous logic
    - Constraints and proper setup/hold fixings
From Repos To Netlist

Source → Bender → Morty → ??? → Yosys

- Bender
- Morty
- ???
- Yosys

Project checkout

HDL pre-processing

Single compile context

Simple HDL

github.com/pulp-platform/bender

github.com/pulp-platform/morty
Our Experiences Synthesizing Iguana

• First challenge: RTL
  • How to bring SystemVerilog into Yosys?
  • PULP codebase is spec-conform
  • As few manual fixes as possible
    • Better tools, not worse code!
UHDM-Yosys

- CHIPS Alliance
- Surelog SystemVerilog parser
- Yosys plugin to read UHDM
- UHDM as an intermediate model
  - Similar: LLHD or LLVM IR
  - Simplification possible at UHDM level
  - One parser for all tools

github.com/chipsalliance/surelog
UHDM-Yosys: First Approach

- Many loud errors
  - Helpful error messages
  - RTL simplification or improving the parser
  - Time consuming

ERROR: 2nd expression of procedural for-loop is not constant
ERROR: Couldn't find search elem: resp in struct
ERROR: Failed to detect width of signal access
ERROR: Failed to detect width of memory access
ERROR: Signal `\Exp_a_D` with non-constant width
ERROR: Latch inferred for signal ... from always_comb process
ERROR: Failed to detect width for identifier
ERROR: Failed to resolve identifier `\EQ` for width detection
ERROR: Invalid bit-select on memory access
ERROR: Don't know how to detect sign and width for ...
ERROR: Don't know how to detect sign and width for ...
ERROR: Assert 'children[0]->type == AST_WIRETYPE' failed in ...
ERROR: Unsupported node type: AST_WIRE
ERROR: enum item already exists
....
Segmentation Fault
....
UHDM-Yosys: First Approach

- **Many loud errors**
  - Helpful error messages
  - RTL simplification or improving the parser
  - Time consuming

- **More severe problems appeared**
  - Scope resolution issues

```verilog
package base_pkg;
    localparam logic BASE = 1'b0;
    localparam logic USED = BASE | 1'b0;
endpackage

package error_pkg;
    localparam logic DERIVED = base_pkg::USED;
endpackage

module top import error_pkg::*; #();
    // ERROR: BASE implicitly declared?
endmodule
```
UHDM-Yosys: First Approach

- Many loud errors
  - Helpful error messages
  - RTL simplification or improving the parser
  - Time consuming

- More severe problems appeared
  - Scope resolution issues
  - Silent errors: extremely hard to detect
  - UHDM is hard to verify
  - Fix Surelog or SV-Plugin?

Too much effort for all PULP IPs

package base_pkg;
  localparam logic BASE = 1'b0;
  localparam logic USED = BASE | 1'b0;
endpackage

package error_pkg;
  localparam logic BASE = 1'b1;
  localparam logic DERIVED = base_pkg::USED;
endpackage

module top import error_pkg::*; #();
  // ERROR: DERIVED is 1'b1 (from BASE)!
endmodule
SV2V: Second Approach

- **SV2V**
  - Translation to Verilog 2005
  - Yosys can read the output
  - Symbolic resolution of the parameters

- **Does not support the entire SV spec**
  - Helpful error messages
  - Haskell requires niche knowledge to extend

- **Symbolic parameter resolution**
  - Very difficult for humans to comprehend
  - Emitted code is hard to simulate/trace

```
localparam [31:0] NumLevels = $unsigned($clog2(NumIn));
wire [[[((2 ** NumLevels) - 2) >= 0 ? (((2 ** NumLevels) - 1) * 91) - 1 : ((3 - (2 ** NumLevels)) * 91) + (((2 ** NumLevels) - 2) * 91) - 1)):(((2 ** NumLevels) - 2) >= 0 ? 0 : ((2 ** NumLevels) - 2) * 91)] data_nodes;
```
SVase: Third Approach

- Slang has a perfect SV score
  - Can parse and elaborate the code
  - Spec-conform, excellent linter, rock-stable, very fast
  - So far: Not much is done with the elaborated AST

- Idea: We simplify SV using Slang: **SVase**
  - Can parse and elaborate the entire PULP codebase
  - Only simplify unsupported constructs -> Simple SV
  - Uniquification, generate resolution, parameter elaboration
  - Simplification is implemented in passes
    - Easy-to-extend (C++) framework

```verilog
localparam [31:0] NumLevels = $unsigned($clog2(NumIn));
localparam int NumLevels = 4;
logic [81:0] data_nodes;
```
module test #( 
  parameter int NoDefaultParam, 
  parameter type PortTypeParam = logic 
  ) () ;
extendmodule

module top #() () ;
  localparam int TopParam = 32'd5 ;
  localparam type TypeParam = struct packed {...};
test #{
  .NoDefaultParam(TopParam), 
  .PortTypeParam(TypeParam)
  } i_test () ;
extendmodule
module test__1699695787177254841 #( // unique parametrization
    ) ();
endmodule

module top__6142509188972423790 #()();
    localparam int TopParam = 32'd5;
    localparam type TypeParam = struct packed {...};
test__1699695787177254841 #( .NoDefaultParam(TopParam),
    .PortTypeParam(TypeParam)
) i_test ();
endmodule
module test__1699695787177254841 #(
  parameter int NoDefaultParam=32'd5,
  parameter type PortTypeParam=struct packed {...}
) ()
endmodule

module top__6142509188972423790 ()
localparam int TopParam = 32'd5;
localparam type TypeParam = struct packed {...};
test__1699695787177254841 #(
  .NoDefaultParam(TopParam),
  .PortTypeParam(TypeParam)
)i_test ();
endmodule
Combination: Final Approach

• **SVase is not completed yet**
  • Question of time: Implement **complex** steps
  • **SV2V** is currently required to translate the rest
  • We are open for **collaboration**!

• **Ideally:**
  • Integrate Slang as a **plugin** into Yosys
  • **Native** SystemVerilog support
Our Experiences Synthesizing Iguana

• **First challenge: RTL**
  - How to bring SystemVerilog into Yosys?
  - PULP codebase is spec-conform
  - As few manual fixes as possible
    • Better tools, not worse code!

• **Second challenge: Synthesis**
  - Yosys struggles with *some modules*
    • *High* memory footprint, *long* synthesis time
    • Reduced quality of results
  - *Early-on* understanding of *timing* is missing

FSiC2023, Paris, July 2023
Yosys Synthesis

Elaborate  Optimize  Generic  Optimize  DFF map  Tech abc

$div  $shift
$add  $memory

$MUX  $MUX  $NOR
$MUX  $AOI
$NOR  $DFF  $DFF

LSOracle >__

abc >__
There Is Something Going Wrong

- A few modules behave weirdly

- **Area/timing suboptimal**
  - Unexpectedly large for their logic complexity
  - Compared to rest of modules: very long paths
  - Looks like a cancer growing on Iguana

- **Explosion in required compute resources**
  - **Scoreboard** alone: >300 GiB RAM required
  - Yosys keeps all synthesis state in memory
  - Very long synthesis time

Very hard to manage!
Two Problems

- **Elaboration**
  - At this stage: no timing information
  - Globally fixed implementations
  - Arithmetic units are slow

- **Part-select issue (#3833)**
  - CVA6 scoreboard
  - Structs not POW2 data types
  - At elaboration: infer $shiftx$
  - Shift by $N$ (keep it generic)
    - Implemented by log shifter and select
    - Stages cannot be optimized
  - Fix: Detect this fixed-stride case
Conclusion

- **Yosys can synthesize Iguana**
  - **SVase + SV2V RTL pre-processing**
  - Iguana is a **complex pipe cleaner** for the flow

- **Yosys can be improved for complex designs**
  - **Slang-Yosys** plugin or frontend
    - Full **SystemVerilog** support
  - **Timing** information during elaboration
  - More **variants** to select
    - **Library of Arithmetic Units?**
  - **Efficient** part-select
  - More **manageable** memory footprint
    - Option to store temporary synthesis data on **disk**

[Link to the full system Verilog support](guest.iis.ee.ethz.ch/~zimmi/arith_lib.html)

**FSiC2023, Paris, July 2023**
Collaboration

• Design and flow will be open ASAP
  • Some cleanup required (SRAM, IOs)
  • A testcase for many open tools
  • We can increase the complexity if required

• We are open for collaboration
  • SVase extension
  • Improvements to Yosys
  • OpenRoad
    • STA & constraints / Timing fixing / DRV fixing
  • Verification
    • Logic equivalency check / post-layout simulation / DFT

github.com/pulp-platform/iguana