Columbus Joint Chapter
SSC37 / CAS04

Presentation will begin at 6:15 EST
Columbus Section Joint Chapter, SSC37/CAS04

- Founded on June 30, 2020
- Technical talks and workshops hosted year-round
- Previous talks covering a myriad of topics posted on our website

TinyTapeout workshop series and shuttle-run sponsorship
Chapter Application: 9/1/24
Tapeout: 11/8/24

SSCS sponsored Arduino hardware development contest!
Completion Deadline: 8/1/24
1: Dr. Suat Ay
   • Professor at the University of Idaho since 2007
   • 10+ years of industry experience at Photobit/Micron imaging

2: Dr. Marvin White
   • Professor at the Ohio State University
   • Inventor of Correlated Double Sampling (CDS)

3: Dr. Paul McManamon
   • President and CTO at Exciting Technology
   • Former Chief Scientist at AFRL
SSCS Educational Chapter of the Year Award

• Our chapter was lucky enough to win the Best Educational Program Award

• Our chapter chair Ramy Tantawy accepted the award at the SSCS flagship conference (ISSCC)

• The chapter was awarded $2000 in total for continued workshops and outreach

Have an idea for a talk or workshop?
• Contact us!
• columbus.sscs.cas@gmail.com
New Officer Announcement!

• Dr. Shane Smith has joined the officer team as the Vice Chair!
  • IC design, test, packaging, and integration subject matter expert

• Vice chair responsibilities
  • Outreach engagement
  • Sponsorship coordination

• Stay tuned for an exciting workshop over the summer from Dr. Smith on PCB design!

https://r2.ieee.org/columbus-ssccas/officers/
CASS Outreach Program 2024

• Foundations of Mixed-Signal IC Design: A Practical Approach to Lab-to-Fab” series
  • Fundamental building blocks in Analog/Mixed Signal SoC’s lectures
  • Tiny Tapeout workshops
  • Printed circuit board design and Arduino workshop
Opportunities for Engagement

Starter Kits for <$100!

S$150*-300 for PCB & ASIC!

* Early-bird special

WE CAN SPONSOR YOU!
Scan this QR code to fill out the form to participate!

Sponsorship Application
https://forms.gle/ypWKDA4zrj8zKAR9A

Event Feedback Form
https://forms.gle/q5SsnPgFxGNzazYM8
Looking for Sponsors

• Our chapter is always looking for sponsorship to increase our educational and outreach footprint.

• If you are interested in sponsoring our chapter at any contribution level, please reach out to the leadership directly or at the below email.

columbus.sscs.cas@gmail.com
CASS New Member Initiative

- Free membership until 08/2024!
- IEEE membership required in good standing
- CASS membership add-on accessible from IEEE account
- CASS membership only $11 for students after 08/2024

Have an idea for a talk or workshop?
- Contact us!
- columbus.sscs.cas@gmail.com

JOIN IEEE CASS
https://ieee-cas.org/

Celebrate our 75th Anniversary with a Complimentary IEEE Circuit and Systems Society Membership for 2024

Who We Are
The IEEE Circuits and Systems Society is the leading organization that promotes the advancement of the theory, analysis, design, tools, and implementation of circuits and systems. The field spans their theoretical foundations, applications, and architectures, as well as circuits and systems implementation of algorithms for signal and information processing.

Offer Details
To take advantage of this offer, simply sign in with your IEEE Account. The membership will present in the Cart at US$0 (This offer is available for a limited time only, and does not apply to the Preferred Package membership).

*This offer is valid until the 2024 IEEE membership year, ending in August 2024.
Eligibility Criteria:
- Must have an IEEE membership,
- New IEEE CASS members, who have never been members before

Membership Benefits
This special offer is limited and provides you with all CASS member benefits at the Essential package, including free online access to our financially sponsored journals and conference proceedings in the IEEE Xplore Digital Library, as well as:

- Discounted registration fees for all nine CASS flagship and premier conferences
- Access a wide variety of educational resources including the CASS Resource Center, CASS Microlearning Program (CASS MiLe), and the CASS-Wide Webinar Series
- Networking with over 100 Chapters worldwide
- Access to 17 Technical Committees and Special Interest Groups as well as 5 CASS Standards Activities Sub-Groups
More Opportunities for Open-Source Hardware Development: PICO

- “Platform for IC Design Outreach”
- Deadline is next Friday, May 10th!
- Similar structure w/ SKY130/GF180 open-source PDKs
- Analog layouts designed and generated in the OpenFASoC [1] environment
- Chaired by Boris Murmann, former Stanford professor and mixed-signal design expert

https://sscs.ieee.org/about/tc-ose/sscs-pico-design-contest
https://www.youtube.com/watch?v=OD27E138uJo

SSC37/CAS04
Columbus Joint-Chapter Seminar
A Practical Approach to Lab-to-Fab series: Tiny Tapeout Workshop (1)

Webinar Location: SenseICs Corporation @ Rev1 Ventures Main Conference Room
Speaker: Sam Ellicott
About Me

- IEEE Columbus SSCS/CAS Vtools coordinator
- PhD Student at The Ohio State University
  - Circuits Laboratory for Advanced Sensors and Systems
  - RF and Mixed-Signal Integrated Circuit (IC) Design
  - True Random Number Generators
- BSEE at Cedarville University (2019)
- Intern at Analog Devices
Workshop Series Goals

• Understand the workflow for open-source tools
  • Demystify the steps required to generate digital designs
• Hands-on introduction to digital design
  • Crash course to Verilog
  • Ability to design/test simple modules
  • Make a simple project
• Have fun!
Workshop Series Outline

• Workshop 1: Introduction
  • Today

• Workshop 2: Tooling and Series Project
  • September/October

• Workshop 3: Design Review
  • October/November
Workshop Series Outline

• Workshop 1: Introduction (tonight)
  • What are Integrated Circuits (ICs)
  • Brief History of ICs
  • Introduction to Digital Design
  • Introduction to Tiny Tapeout

• Workshop 2: Tooling and Series Project

• Workshop 3: Design Review
Formatting Note

• Some slides are to aid understanding
  • Provide background knowledge that I think is interesting
  • Not required for day-to-day design work

• Background information slides
  • Title in red

• Terminal commands
  • Commands are in bold italic
Introduction to Integrated Circuits
Hands up if

- You’ve used an app
- You’ve written a computer program
- Used an Arduino (or similar)
- Designed a chip
- Had your own chip manufactured
- Manufactured your own chip
Some Definitions

- What is an integrated circuit?
  - Also called IC or chip
  - Multiple transistors all in the same substrate
  - Connections between transistors

- What is a transistor?
  - For digital circuits: a electronic controllable switch
  - Things get more complicated for analog circuits
Why do we care?
A modern IC (Apple M1 Pro)

- 33.7 Billion Transistors
- $245 \text{mm}^2 \rightarrow 16 \text{mm}$ square
- $5\text{nm}$ process node

https://www.anandtech.com/show/17019/apple-announced-m1-pro-m1-max-giant-new-soCs-with-allout-performance
A Sense of Scale

Powers of ten
Charles and Ray Eames.
A Sense of Scale

Powers of ten
Charles and Ray Eames.
Extreme Ultraviolet Lithography
History of Integrated Circuits
Before Transistors: 1944

Electronic Numerical Integrator and Computer: programmable for artillery calculations
The First Transistor: 1947
Discrete Transistor

Integrated Circuits: 1958

Jack Kilby 1958 at Texas Instruments
Integrated Circuits: 1959

Robert Noice at Fairchild Semiconductor

Shrinking Transistors and Technologies

1950s
Silicon Transistor
1 Transistor

1960s
TTL Quad Gate
16 Transistors

1970s
8-bit Microprocessor
4500 Transistors

1980s
32-bit Microprocessor
275,000 Transistors

1990s
32-bit Microprocessor
3,100,000 Transistors

2000s
64-bit Microprocessor
592,000,000 Transistors

2010s
3072-Core GPU
8,000,000,000 Transistors

100 um 10 um 1 um 100 nm 10 nm
Moore’s Law

- Number of transistors doubles every 2 years
- Exponential growth!
Moore’s Law: TSMC Processes

Tiny Tapeout uses a 0.13 μm process

FinFET Transistors

Apple M1 uses this process

https://www.tsmc.com/english/dedicatedFoundry/technology/logic/L_5nm
Transistor Evolution

- Planar Transistors: > ~20nm
- FinFET Transistors: < ~20nm
- Gate-All-Around Transistors: Future
- Video about FinFET Transistors
- Video about GAA Transistors

Power Density

Pentium 4 (Northwood)
55M 130nm Transistors
~50W TDP

https://hothardware.com/photo-gallery/newsitem/53364?image=big_amd_die_map.jpg&tag=popup

Ryzen 5800X
4.15B 7nm Transistors
~65W TDP
Dennard Scaling

What if Moore's law/Dennard Scaling worked for cars?

<table>
<thead>
<tr>
<th>Year</th>
<th>Speed (mph m/s)</th>
<th>Fuel Efficiency (mpg)</th>
<th>Moore's Car Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>170 mph (75 m/s)</td>
<td>14.3 mpg</td>
<td>80,000 million m/s</td>
</tr>
<tr>
<td>2020</td>
<td>300 mph (134 m/s)</td>
<td>38.8 mpg</td>
<td>15,000 million mpg</td>
</tr>
</tbody>
</table>

Further Information

- This is just scratching the surface of the history here

- Professor Marvin White’s CASS Talk

- PBS Documentary on Silicon Valley

- Asianometry Youtube Channel
Introduction to Digital Design
What are we trying to do?

- Concept (algorithm, behavior, etc) → Physical design (gates)
How does this work?

- Calculate Nth Fibonacci number
- Software (C program)

```c
int fib(int n) {
    int a = 0, b = 1, c = 0, i;
    if (n == 0) {
        return a;
    }
    for (i = 2; i <= n; i++) {
        c = a + b;
        a = b;
        b = c;
    }
    return b;
}
```

https://godbolt.org/z/dPKEe7qE3
How does this work?

- Calculate Nth Fibonacci number
- Hardware (Digital Logic)

**Hardware Description Language (Verilog)**

```verilog
reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;
assign o_busy = (iteration != RESET);
assign o_fib = current;
always @ (posedge i_clk) begin
  if (!o_busy & !i_stb) begin
    iteration <= _i_n;
    prev <= [WIDTH-1:0] 1;
    current <= [WIDTH-1:0] 0;
  end
  else if (o_busy) begin
    iteration <= iteration - ONE;
    current <= prev + current;
    prev <= current;
  end
  if (i_reset) begin
    prev <= 0;
    current <= 0;
  end
endmodule
```

**Synthesis Tool (Yosys)**

**Generic Logic Gates**
What is a “gate”?  

- “Legos” of digital design

- Combinational Logic Gate  
  - Performs digital logic function  
  - AND, OR, NOT, ...

- Sequential Logic  
  - Memory Element  
  - Flip-Flops  
  - Stores data for a “clock cycle”

https://github.com/sellicott/inkscape-figures
What is Verilog? (Wires and Registers)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

*There is also gate level verilog

Define Signals
What is Verilog? (Wires and Registers)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

*There is also gate level verilog

```verilog
// global control signals
input wire i_reset;
input wire i_clk;

// control signals
input wire i_stb;
output wire o_busy;

// module io
input wire [WIDTH-1:0] i_n;
output wire [WIDTH-1:0] o_fib;

reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;
```

Define Signals
What is Verilog? (Assign Statements)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

Define Signals

Purely Combinational Logic

```verilog
// global control signals
input wire i_reset;
input wire i_clk;

// control signals
input wire i_stb;
output wire o_busy;

// module io
input wire [WIDTH-1:0] i_n;
output wire [WIDTH-1:0] o_fib;

reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;

assign o_busy = (iteration != RESET);
assign o_fib = current;
```
What is Verilog? (Always Blocks)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

```verilog
// global control signals
input wire i_reset;
input wire i_clk;

// control signals
input wire i_stb;
output wire o_busy;

// module io
input wire [WIDTH-1:0] i_n;
output wire [WIDTH-1:0] o_fib;
reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;
assign o_busy = (iteration != RESET);
assign o_fib = current;
```

- Define register behavior
  - Write to registers in “always” block
  - Defines what the “next state” of the register should be
  - Only evaluated on rising edges
  - Multiple Always blocks are executed in parallel

```verilog
always @(posedge i_clk)
begin
  if (io_busy or i_stb) begin
    iteration <= i_n;
    prev [WIDTH-1:0] <= 1;
    current [WIDTH-1:0] <= 0;
  end
  else if (o_busy) begin
    iteration <= iteration - ONE;
    current <= prev + current;
    prev <= current;
  end
  if (i_reset) begin
    iteration <= RESET;
    prev [WIDTH-1:0] <= 1;
    current [WIDTH-1:0] <= 0;
  end
end
```

Define Signals

Purely Combinational Logic

Sequential Logic Behavior
What is Verilog? (Always Blocks)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

Define Signals

Purely Combinational Logic

Sequential Logic Behavior

- Always blocks evaluation
  - Top to bottom
  - <= assignments happen in parallel

- Statements lower in block have higher priority

- Can feel like “normal” programing
  - *Don’t get complacent*
What is Verilog? (Always Blocks)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

```verilog
// global control signals
input wire i_reset;
input wire i_clk;

// control signals
input wire i_stb;
output wire o_busy;

// module io
input wire [WIDTH-1:0] i_n;
output wire [WIDTH-1:0] o_fib;
reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;

assign o_busy = (iteration != RESET);
assign o_fib = current;
```

*Always blocks may also contain logic

Sequential Logic Behavior

Purely Combinational Logic
What is Verilog? (Always Blocks)

- Hardware Description Language
  - Behaviorally* describes the gates we want to generate
  - Combinational and sequential gates generated from code

```verilog
// global control signals
input wire i_reset;
input wire i_clk;

// control signals
input wire i_stb;
output wire o_busy;

// module io
input wire [WIDTH-1:0] i_n;
output wire [WIDTH-1:0] o_fib;
reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;
assign o_busy = (iteration != RESET);
assign o_fib = current;

always @(posedge i_clk) begin
  if (i_busy & i_stb) begin
    iteration <= i_n;
    prev [WIDTH-1:0] <= 1;
    current [WIDTH-1:0] <= 0;
  end
  else if (i_busy) begin
    iteration <= iteration - ONE;
    current <= prev + current;
    prev <= current;
  end
end
```

Define Signals

Purely Combinational Logic

Sequential Logic Behavior

- Always blocks may also contain logic
A simple example

• How does the system evaluate over time?
  • What is the state of A’, B’, Y, and Z?

https://wavedrom.com/
A simple example

• How does the system evaluate over time?
  • What is the next state of A', B', Y, and Z?

https://wavedrom.com/
A simple example

- How does the system evaluate over time?
- What is the next state of A', B', Y, and Z?
A simple example

- How does the system evaluate over time?
  - What is the next state of A', B', Y, and Z?

https://wavedrom.com/
A simple example

- How does the system evaluate over time?
- What is the next state of A', B', Y, and Z?

https://wavedrom.com/
Real Logic Gates

- How is a NOR gate built?
- Transistors → Switches

[Diagram of NOR gate with PMOS and NMOS transistors]
Standard Cell Library

- Skywater provides designs for a large number of logic gates
- Standard Cell Library

> 5x larger

Sky130 Stackup

- World’s first open source manufacturable PDK
- Profile of layers in design
- Gates are connected on metal2 – metal5
What’s the difference between Software and Hardware?

**Software**

- \(\text{fib:}\)
  - \(\text{beg a0,zero,.L4}\)
  - \(\text{li a5,1}\)
  - \(\text{ble a0,a5,.L5}\)
  - \(\text{addi a3,a0,1}\)
  - \(\text{li a5,2}\)
  - \(\text{li a0,1}\)
  - \(\text{li a4,0}\)

- \(\text{.L3:}\)
  - \(\text{mv a2,a0}\)
  - \(\text{addi a5,a5,1}\)
  - \(\text{add a0,a0,a4}\)
  - \(\text{mv a4,a2}\)
  - \(\text{bne a5,a3,.L3}\)
  - \(\text{ret}\)

- \(\text{.L4:}\)
  - \(\text{li a0,0}\)
  - \(\text{ret}\)

- \(\text{.L5:}\)
  - \(\text{li a0,1}\)
  - \(\text{ret}\)

**Hardware**

- Software executes sequentially
- Hardware is “always running”
  - Copies run in parallel

[Diagram of hardware circuit]

[Diagram of software code]

https://www.pngall.com/versus-png/download/54595
Recap: Digital Design

- Concept (algorithm, behavior, etc) → Physical design (gates)

```verilog
reg [WIDTH-1:0] iteration;
reg [WIDTH-1:0] prev;
reg [WIDTH-1:0] current;
assign o_busy = (iteration != RESET);
assign o_fib = current;
always @(posedge i_clk) begin
    if (i_reset || (o_busy && i_stb)) begin
        iteration <= i_n;
        prev [WIDTH-1:0] <= 1;
        current [WIDTH-1:0] <= 0;
    end
    else if (o_busy) begin
        iteration <= iteration - ONE;
        current <= prev + current;
        prev <= current;
    end
    if (i_reset) begin
        iteration <= RESET;
        prev [WIDTH-1:0] <= 1;
        current [WIDTH-1:0] <= 0;
    end
endmodule
```

Generic Gates

Standard Cells

Final Design
OpenLane Flow

WOKWi or Verilog Files

Output Design (GDS File)
OpenLane Flow

WOKWi

or

Verilog Files

Elaboration:
Convert to generic gates
OpenLane Flow

WOKWi or Verilog Files

The OpenLane Flow

Synthesis: Convert to “real” logic gates
OpenLane Flow

Place and Route (PnR): Place gates and route wires between them.

Verilog Files

WOKWi

or

Place and Route (PnR): Place gates and route wires between them.
OpenLane Flow

WOKWi or Verilog Files

Signoff:
Make sure everything matches
No rules are violated
OpenLane Flow

WOKWi or Verilog Files

We're mostly dealing with this part

Tiny Tapeout (via Openlane) handles most of this for us!
GDS examples (all 70um x 70um)

- Binary to decimal converter: 25 cells
- 8 bit counter: 49 cells
- 4 bit counter & bcd: 50 cells
What did we learn?

• Design → Digital Flow → Physical Design

• Digital Design
  • Gates are building blocks of the design
  • Combinational Logic/Sequential Logic
  • Verilog lets us describe gates with code

• Digital Flow (OpenLane)
  • Elaboration: Hardware Description Language → Ideal Gates
  • Synthesis: Generating real gates from the design
  • Place and Route: Layout and route physical gates
  • Signoff: Check the design for layout/timing errors
Tiny Tapeout
How Tiny Tapeout Works

- Cloud based design
  - Runs OpenLane in Github actions
  - No tool install or download
  - 3D viewer / explorer
  - Example Design

- ~500 Projects merged into one IC
  - Reduced cost
  - Try other peoples designs
  - Test PCB
How Tiny Tapeout Works

- Cloud based design
- Runs OpenLane in Github actions
- No tool install or download
- 3D viewer / explorer
- Example Design
  - ~500 Projects merged into one IC
- Reduced cost
- Try other peoples designs
- Test PCB
Tiny Tapeout 02 Datasheet

Project Repository
https://github.com/TinyTapeout/tinytapeout-02

December 7, 2022

Contents

Render of whole chip

Projects

0: Test Inverter Project
1: SIMON Cipher
2: HD74480 Clock
3: Scrolling Binary Matrix display
4: Power supply sequencer
5: Duty Controller
6: S4GA: Super Slow Serial SRAM FPGA
Physical Interface

- **1x Tile**
  - 160 x 100\(\mu m\) size
  - \(\sim\)1000 gates
  - Can buy more than 1 tile

- **Pins**
  - Clock (~50MHz)
  - Reset
  - 8x Inputs
  - 8x Outputs
  - 8x Bidirectional

![Tiny-Tapeout Module](image_url)
Wokwi Design Flow

• **Wokwi** by Uri Shaked
  • Online Simulator
  • Exports Verilog Netlist
• **Examples**
  • Padlock
  • UART
  • 7-Segment Display

- dtype flop
- inverter
- 2 input and
- 2 input or
- 2 input xor
- 2 input mux
- 2 input nand
Using GitHub Flow for Verilog

- **Tutorial Video**
- **Example Project** (Fibonacci Sequence Generator)
- Create new GitHub project by using the template
  - The project should be public!
  - **Enable GitHub Pages** (set to GitHub Actions)
- Update `info.yaml`
  - Top module
  - Source files
  - Area
  - Documentation
- Update `docs/info.md`
  - Add detailed documentation for your project
Live Demo

WE'RE DOING IT LIVE!

https://github.com/sellicott/sellicott_fib_seq

If live demo isn't working

Demo Slides

Skip Demo
Demo Slides (Home)
Demo Slides (wrapper source)
Demo Slides (module source)
Demo Slides (info.yaml)
Demo Slides (Testbench)
Demo Slides (Makefile)
Demo Slides (GitHub Actions)
Demo Slides (Test Action)
Demo Slides (Test Results)
Demo Slides (Test GTKWave)
Demo Slides (Docs Action)
Fibonacci Sequence Generator

- Author: Samuel Ellicott
- Description: generate the Nth fibonacci number
- Language: Verilog

How it works

The project takes in the index of Fibonacci number to generate \( n=0 \rightarrow 1, n=1 \rightarrow 1,... \). Where \( n \) is an 8-bit unsigned integer on the \( n[7:0] \) pins. To start generating the sequence start_stb should be asserted for one clock cycle. While the module is working, the busy signal will be asserted. After the busy signal falls to 0, the Nth Fibonacci number is available on \( fib[7:0] \) pins

Pinout

<table>
<thead>
<tr>
<th>#</th>
<th>Input</th>
<th>Output</th>
<th>Bidirectional</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( n[0] )</td>
<td>( fib[0] )</td>
<td>start_stb</td>
</tr>
<tr>
<td>1</td>
<td>( n[1] )</td>
<td>( fib[1] )</td>
<td>busy</td>
</tr>
<tr>
<td>7</td>
<td>( n[7] )</td>
<td>( fib[7] )</td>
<td></td>
</tr>
</tbody>
</table>
Demo Slides (GDS Action)

- **Synthesis/PnR Log Files**
- **Gate Level Test Results**
Demo Slides (Cell Usage)
Demo Slides (2D Render)
Demo Slides (3D Viewer)

KEYS
1: Hide Fill, Decap, Tap cells
2: Hide top cell geometry
3: Isolate mouse over cell
Mouse over: FILLER_0_19_20S (sky130_et_sc_nd_decap_12)

GitHub Actions
GDS Action
Home
2D Render
Demo Slides (Settings)

Set to "GitHub Actions"
Running Locally

• Tiny Tapeout allows for cloud based development
  • Full flow in GitHub actions
  • Only need GTKWave (to view simulation results)
  • Slow iteration loop

• We can run the whole OpenLane flow locally...
  • Large installation size (~10Gb)
  • Slightly involved installation process (on Windows)

• Run simulations locally
  • Develop Verilog locally
  • Smaller (~1.5Gb)
  • Simulate behavioral HDL → Basic functionality
  • Harden with Tiny Tapeout flow (GitHub actions)
Installing Tools

• Simulation Tools
  • [https://github.com/YosysHQ/oss-cad-suite-build](https://github.com/YosysHQ/oss-cad-suite-build)
  • Extract file to a path without spaces
  • GTKWave, Icarus Verilog, Yosis, Etc

• Start by running `start.bat`
  • Should see ‘[OSS Cad Suite]’
  • Move around with ‘cd’
  • Drag and drop folders into terminal
    • Puts folder path into terminal

• Other Tools
  • Git: [https://git-scm.com/downloads](https://git-scm.com/downloads)
  • VS Code: [https://code.visualstudio.com/](https://code.visualstudio.com/)
Local Simulation: Running Tools

- In [OSS CAD Suite] terminal!
  - Type the tool you want to use
  - Use ‘gtkwave’ for GTKWave, etc
Local Simulation: Running Simulator

- Using Icarus Verilog for Simulation
- In [OSS CAD Suite] terminal move to folder with Verilog code
- Elaborate → Simulate design
  - `iverilog -o tb.vvp -DICARUS <Verilog file 1> <Verilog file 2> ...`
  - `vvp tb.vvp`
- Generates `tb.vcd` for GTKWave
- More Information:
  - [https://steveicarus.github.io/iverilog/usage/getting_started.html](https://steveicarus.github.io/iverilog/usage/getting_started.html)
Local Simulation: Testbench

- Define what file to write simulation to
  - $dumpfile sets the filename
- Select what data to save
  - $dumpvars selects what signals to save
- See here

- Define variables that we can interact with
  - Registers for module inputs
  - Wires for module outputs

- Instantiate our Tiny Tapeout module
  - Hook up power and gnd if doing a gate level test (Run after synthesis)
  - Other ports hooked up to variables
Local Simulation: Testbench

**tb.v Testbench**

- Only run this code if ICARUS flag is set
  - Avoid breaking GitHub Action sims
- Define clock to run at 50MHz
- Run tests
  - Three segments
    - Initialzation
    - Tests (first 10 fibonacci numbers)
    - Closing simulation
**Local Simulation: Testbench**

```verilog
// task to initialize all of the top level registers and reset the system

task initialize();
begin:
    $display("Start");
    #2;
    $display("Reset");
    ena = 1'h1;
    ui_in = 8'h0;
    uio_in = 8'h0;
    rst_n = 1'h0;
    // wait 10 clock cycles
    repeat(10) @(posedge clk);
    rst_n = 1'h1;
endtask

// task close();
begin:
    $display("closing");
    // wait 10 clock cycles to give us time to read last value
    repeat(10) @(posedge clk);
$finish;
endtask
```

- Puts the system into a defined state
- Sets the values for all the input registers
- Reset the system
- Delay a bit before ending the simulation
- Makes it easier to read waveforms

$\texttt{finish}$ closes the simulator
Local Simulation: Testbench

```verilog
task test_n_fib(
    input integer n
);
begin: test n fib block
    integer fib_idx;
    fib_idx = 0;
    for (fib_idx = 0; fib_idx < n; fib_idx++)
        begin: fib test iter
            integer fib_calc;
            integer fib_hw;
            fib_hw = 0;
            // calculate the current fibonacci sequence value
            fib_calc = calc_fib(fib_idx);
            run_fib_seq(fib_idx);
            fib_hw = run_fib_seq;  
            $strobe("Test n=%d", fib_idx);
            $strobe("hw fib: %d, sw fib: %d", fib_hw, fib_calc);
            // exit if the values don't match  
            if(fib_hw != fib_calc) close();
        end
    end
endtask
```

- For loop to test the $n$th Fibonacci sequence number
- Weird variable definition to keep Verilog happy
- Calculate the Fibonacci number using a “software” method
- Calculate again using module
- Exit simulation if values don’t match
Local Simulation: Testbench

- Assert the `i_stb` pin and set what `n` to test
- Wait a clock cycle and clear the `i_stb` pin
- Wait at least one clock cycle
- Wait until `o_busy` is cleared
Local Simulation: Testbench

```verilog
// iteratively calculate the n-th Fibonacci sequence number
function Integer calc_fib(
    input Integer n
);
begin: calc_fib_block
    Integer a;
    Integer b;
    Integer c;
    Integer i;
    a = 0;
    b = 1;
    c = 0;
    if (n == 0) begin
        calc_fib = a;
    end
    else begin
        for (i = 1; i < n; i++) begin
            c = a + b;
            a = b;
            b = c;
        end
        calc_fib = b;
    end
endfunction
```

- Calculate the n-th Fibonacci number
- Do the whole calculation in Verilog
- Can write basically like software
Expected Results

- Assert `i_stb` for a clock cycle
- `o_busy` high until done calculating
- Result on `o_fib`
Expected Results

- Assert `i_stb` for a clock cycle
- `o_busy` high until done calculating
- Result on `o_fib`
Next Steps

• Install the simulation tools locally
• Follow some digital design tutorials
• Follow some Verilog tutorials
  • NAND Land Verilog tutorial
  • Reference Designer Verilog tutorial
• Test your own designs
Step by Step Instructions

• Goal: Recreate Fibonacci Sequence Generator

• Install Tools
  • Git: https://git-scm.com/downloads
  • VS Code: https://code.visualstudio.com/
  • OSS-CAD-Suite: https://github.com/YosysHQ/oss-cad-suite-build

• Setup Tiny Tapeout Project
  • Make your own project based on the template
  • Enable GitHub Pages (set to GitHub Actions)
  • Update info.yaml file

• Write Verilog Code
  • Implement Fibonacci Sequence state machine
  • Write testbench
  • To learn Verilog syntax: Verilog tutorials

• Simulate Design
  • Use Icarus Verilog to simulate testbench
  • Use GTKWave or VaporView to view output waveforms

• Push Code to GitHub
Where To Get Help

• Tiny Tapeout Discord
  • Lots of people with deeper knowledge than me
  • I’m there too (@sellicott)

• Learning Resources
  • https://tinytapeout.com/digital_design/
  • https://tinytapeout.com/hdl/
  • https://r2.ieee.org/columbus-ssccas/resources/

• Questions about this workshop
  • Chapter: columbus.sscs.cas@gmail.com

Here!

Tiny Tapeout Discord
https://discord.gg/BspcX9SB2h
Acknowledgements

Matt Venn
• Original workshop slides
• Advice on running a workshop

Everyone who worked on Tiny Tapeout!

https://tinytapeout.com/credits/
Scan this QR code to fill out the form to participate!

Sponsorship Application
https://forms.gle/ypWKDA4zrj8zKAR9A

Event Feedback Form
https://forms.gle/q5SsnPgFxGNzazYM8
This page unintentionally left blank