Open-Source-Chip-Design

A new Playground for Ham Radio?

Prof. Dr.-Ing. Matthias Jung (DL9MJ)

28.06.2025

Start

Raise your hand if:

- You've used an app

Start

Raise your hand if:

- You've used an app
- You've written a computer program

Start

Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary

Start

Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit

Start

Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit
- You've designed a circuit

Start

Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit
- You've designed a circuit
- You've used an FPGA

Start

Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit
- You've designed a circuit
- You've used an FPGA
- You've designed a chip

Start

Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit
- You've designed a circuit
- You've used an FPGA
- You've designed a chip
- You had your own chip manufactured

Start

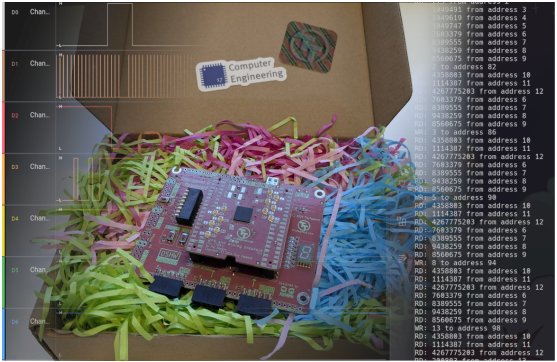
Raise your hand if:

- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit
- You've designed a circuit
- You've used an FPGA
- You've designed a chip
- You had your own chip manufactured
- You've manufactured your own chip

Start

Raise your hand if:

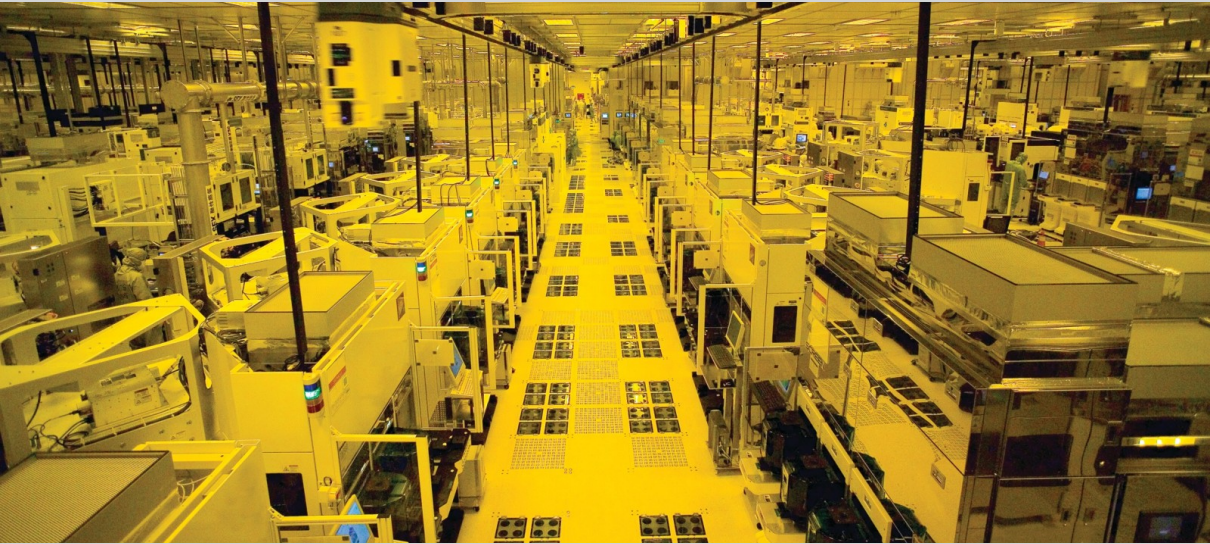
- You've used an app
- You've written a computer program
- You can count to 16 in binary
- You've built a circuit
- You've designed a circuit
- You've used an FPGA
- You've designed a chip
- You had your own chip manufactured
- You've manufactured your own chip



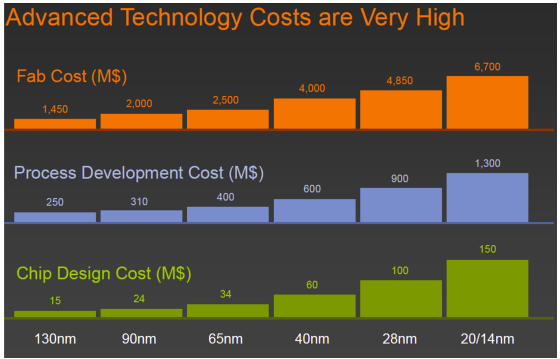
Slides and Code:
https://github.com/CEJMU/tto8-fir

Semiconductor Fabs

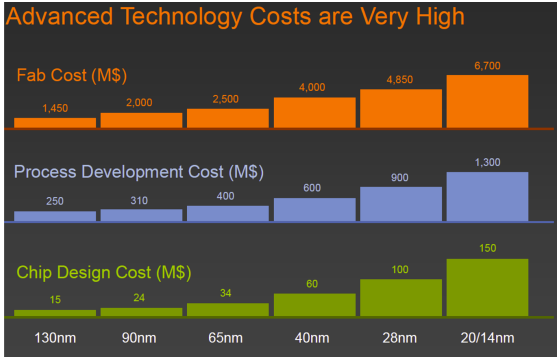
Semiconductor Fabs

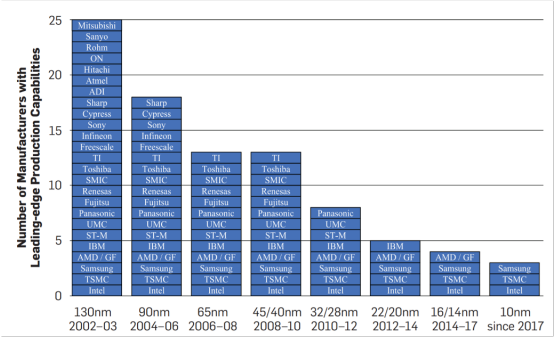


Manufacturing Cost

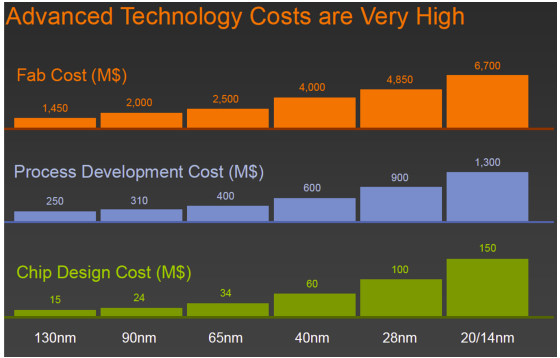


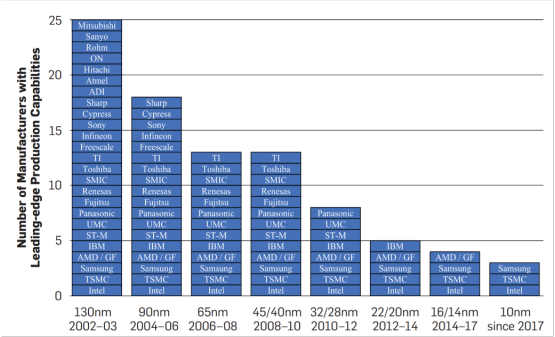
Manufacturing Cost





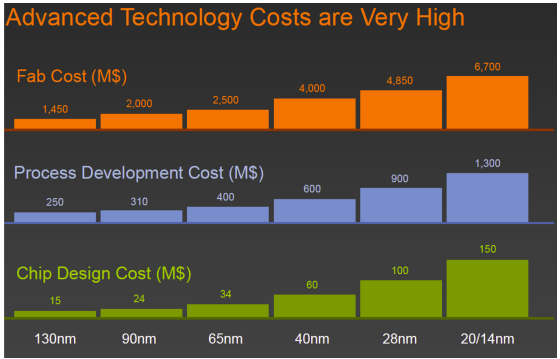
Manufacturing Cost

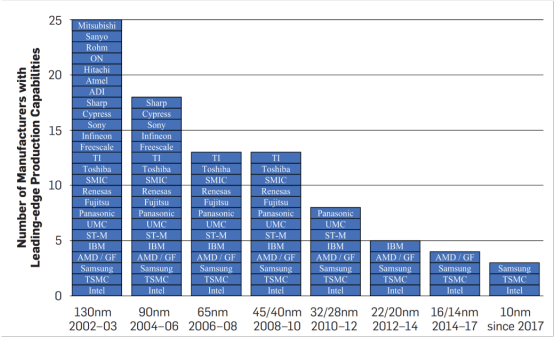




- Advanced nodes are expensive

Manufacturing Cost



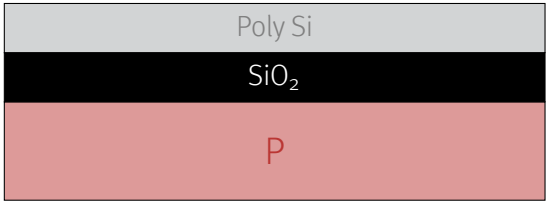


- Advanced nodes are expensive
- Written-off nodes, like 130nm, are affordable, now even for hobbyists (~10k)

Manufacturing Process

Steps

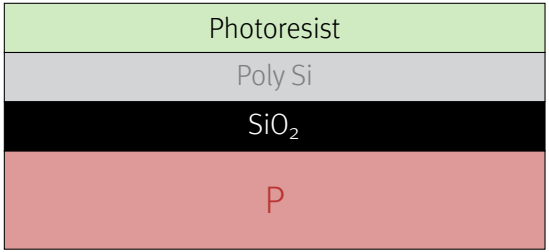
1. Deposition



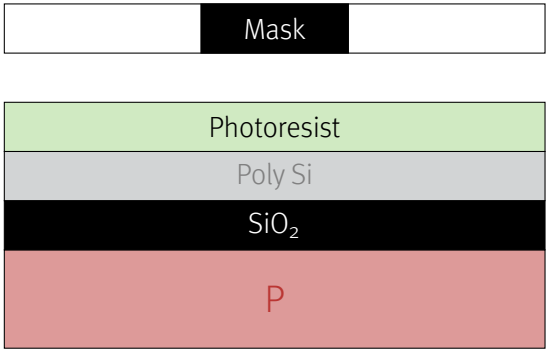
Manufacturing Process

Steps

1. Deposition
2. Litography



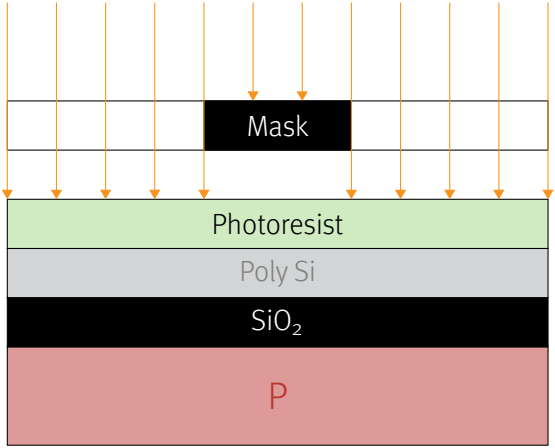
Manufacturing Process



Steps

1. Deposition
2. Litography

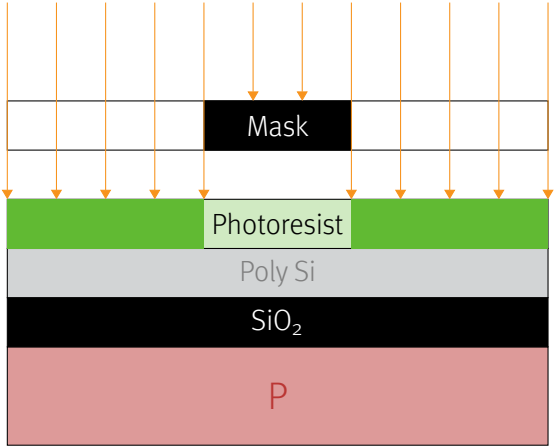
Manufacturing Process



Steps

1. Deposition
2. Litography

Manufacturing Process



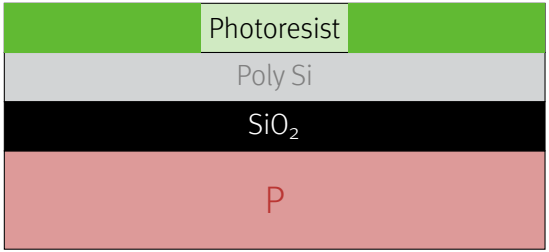
Steps

1. Deposition
2. Litography

Manufacturing Process

Steps

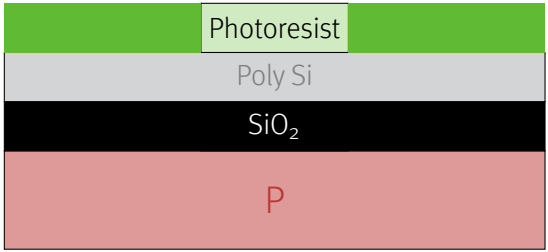
1. Deposition
2. Litography



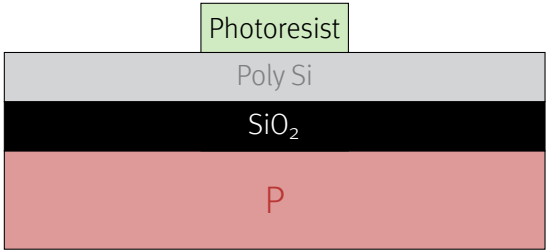
Manufacturing Process

Steps

1. Deposition
2. Litography
3. Etching



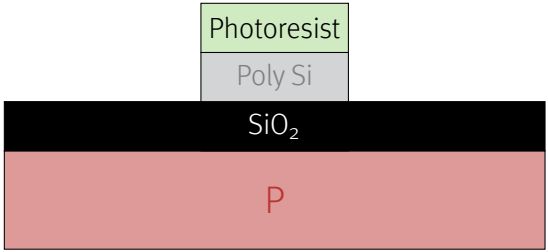
Manufacturing Process



Steps

1. Deposition
2. Litography
3. Etching

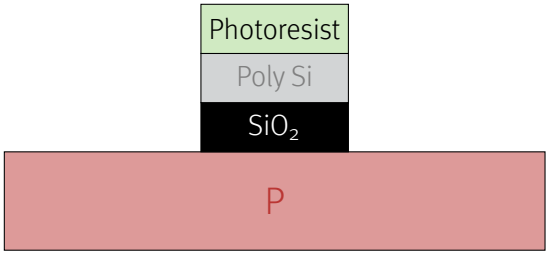
Manufacturing Process



Steps

1. Deposition
2. Litography
3. Etching

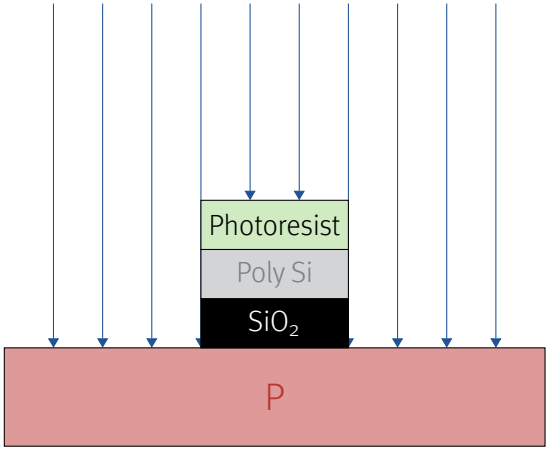
Manufacturing Process



Steps

1. Deposition
2. Lithography
3. Etching
4. Doping

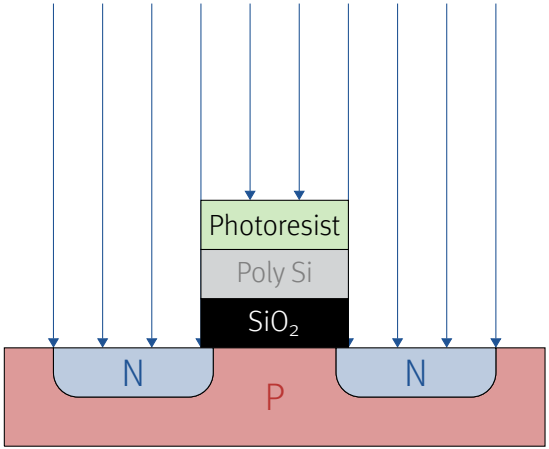
Manufacturing Process



Steps

1. Deposition
2. Lithography
3. Etching
4. Doping

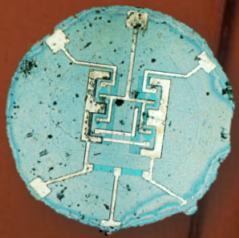
Manufacturing Process



Steps

1. Deposition
2. Lithography
3. Etching
4. Doping

Steps 1-4 can be repeated 400-1000 times.

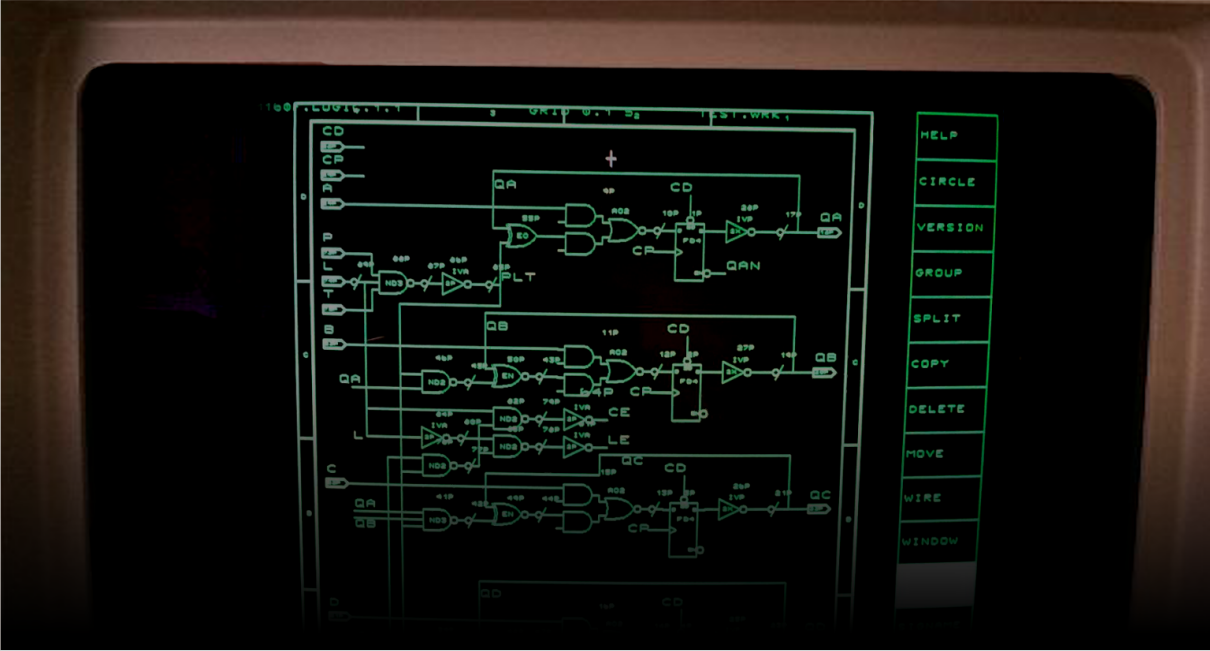


Era of Transistors:

- Manual cutting of Rubylith
- First *Design Automation Conference* DAC (1963)

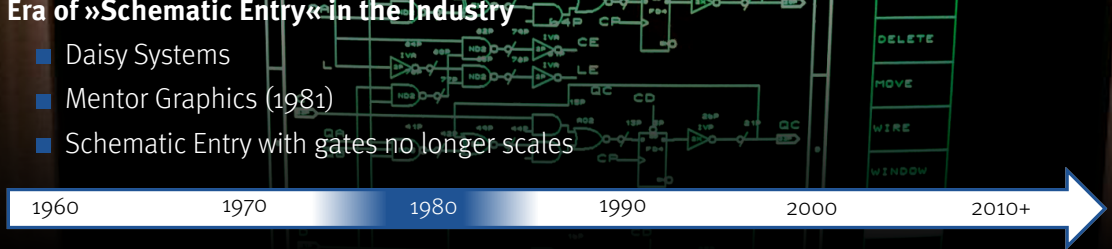
1960 1970 1980 1990 2000 2010+

Source: Computer History Museum, USA


HELP
CIRCLE
VERSION
GROUP
SPLIT
COPY
DELETE
MOVE
WIRE
WINDOW
CD
CP
A
P
L
T
B
QA
QB
QC
QD
CE
LE
PLT
QAN
CR

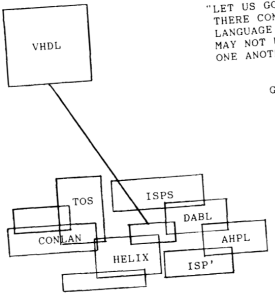
Era of »Schematic Entry« in the Industry

- Daisy Systems
- Mentor Graphics (1981)
- Schematic Entry with gates no longer scales



Quelle: Simon F. (http://www.cpu-ns32k.net)

THE NEED FOR A COMMON LANGUAGE

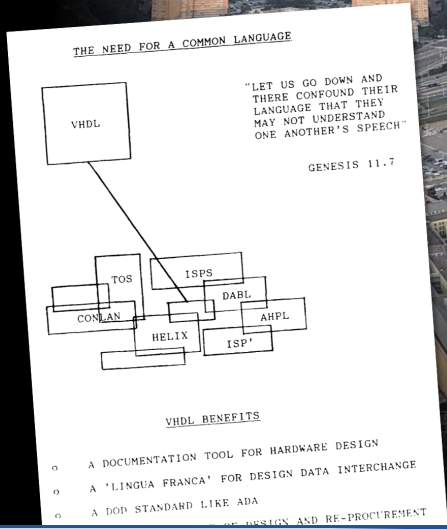


"LET US GO DOWN AND THERE CONFOUND THEIR LANGUAGE THAT THEY MAY NOT UNDERSTAND ONE ANOTHER'S SPEECH"

GENESIS 11.7

VHDL BENEFITS

- A DOCUMENTATION TOOL FOR HARDWARE DESIGN
- A 'LINGUA FRANCA' FOR DESIGN DATA INTERCHANGE
- A DOD STANDARD LIKE ADA
- DOCUMENTATION FOR RE-DESIGN AND RE-PROCUREMENT

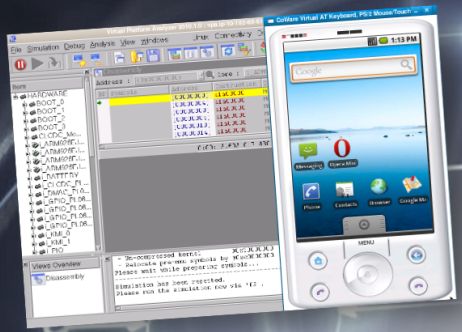


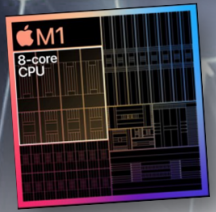
The necessity of a common language

- VHSIC Program by DoD (1981)
- Verilog by Gateway Design Automation (1983)
- VHDL IEEE Standard (1987)

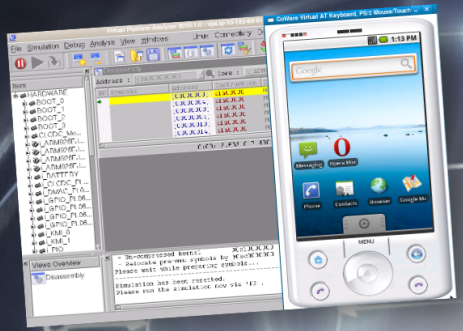
1960 1970 1980 1990 2000 2010+

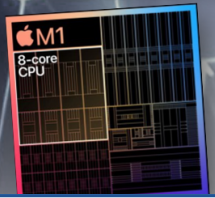
Quelle: VHSIC Annual Report 1987, DoD, Pentagon





RISC-V





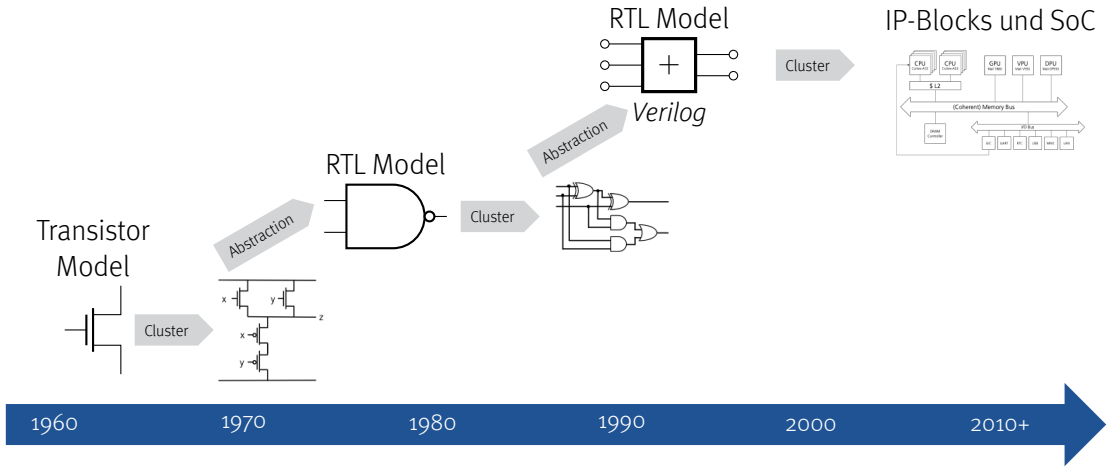
RISC-V

Era of Electronic System Level Design:

- Platform based design, HW/SW codesign
- Systems on Chip (SoC), IP-XACT (2009)
- High Level Synthesis (HLS)
- Virtual Prototyping (2006)
- SystemVerilog (2003), SystemC (2006)
- Open Source HW und Toolchains (2010-2025)

1960 1970 1980 1990 2000 2010+

Design Methodology Over Time



What is Verilog?

Verilog and System Verilog

What is Verilog?

- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)

What is Verilog?

- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Verilog and System Verilog

What is Verilog?

- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Why do we need Verilog?

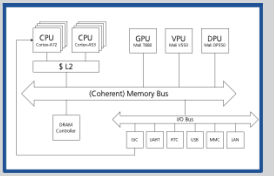
Verilog and System Verilog

What is Verilog?

- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Why do we need Verilog?

1. Doc. & Spec.



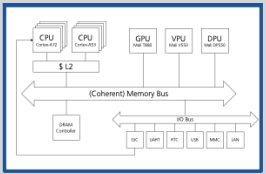
Verilog and System Verilog

What is Verilog?

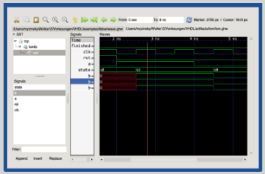
- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Why do we need Verilog?

1. Doc. & Spec.



2. Simulation



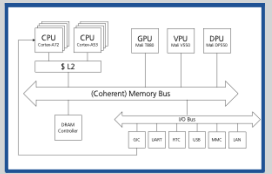
Verilog and System Verilog

What is Verilog?

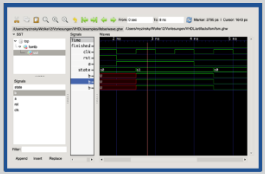
- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Why do we need Verilog?

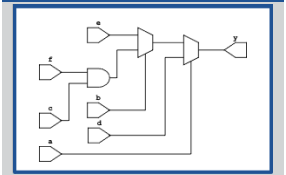
1. Doc. & Spec.



2. Simulation



3. Synthesis



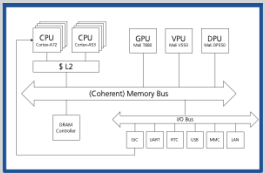
Verilog and System Verilog

What is Verilog?

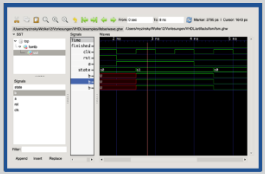
- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Why do we need Verilog?

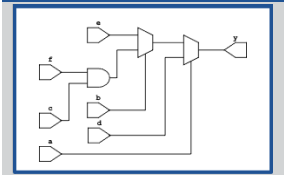
1. Doc. & Spec.



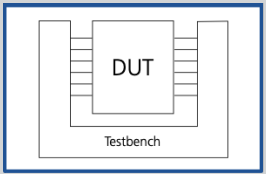
2. Simulation



3. Synthesis



4. Verification



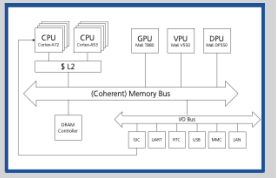
Verilog and System Verilog

What is Verilog?

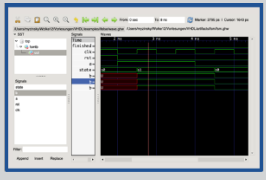
- Hardware description language IEEE 1364 (Verilog) and IEEE 1800 (System Verilog)
- Verilog is a portmanteau of *Verification* and *Logic*

Why do we need Verilog?

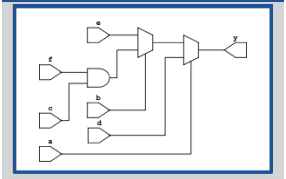
1. Doc. & Spec.



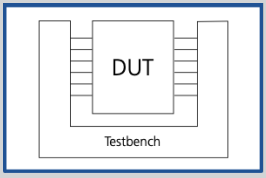
2. Simulation



3. Synthesis



4. Verification



Disclaimer: This is not a SystemVerilog class, if you want one look here on my YouTube-Channel

Verilog is not a Programming Language!

C++

```
1 bool a = 0;
2 bool b = 1;
3 ...
4 a = b;
5 b = a;
```

Verilog is not a Programming Language!

C++

```
1 bool a = 0;
2 bool b = 1;
3 ...
4 a = b;
5 b = a;
```

Verilog

```
1 logic a = 0;
2 logic b = 1;
3 ...
4 a <= b;
5 b <= a;
```

Verilog is not a Programming Language!

C++

```
1 bool a = 0;
2 bool b = 1;
3 ...
4 a = b;
5 b = a;
```

Verilog

```
1 logic a = 0;
2 logic b = 1;
3 ...
4 a <= b;
5 b <= a;
```

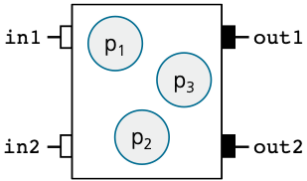
Assignment Operations in Verilog

Blocking (=): Each assignment is executed sequentially (blocking)

Non-Blocking (<=): Assignments are executed in parallel

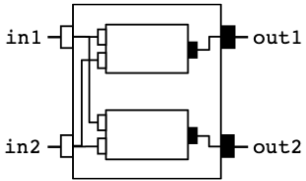
Verilog Modeling Approaches

Behavioral Style



Functional description of behavior using concurrent processes

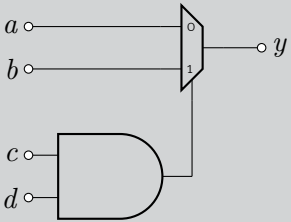
Structural Style



Description of behavior through aggregation of sub-components

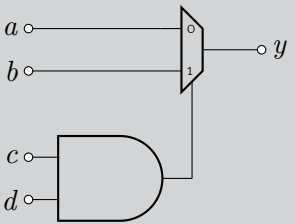
Combinatorial and Sequential Circuits

Combinatorial Circuit

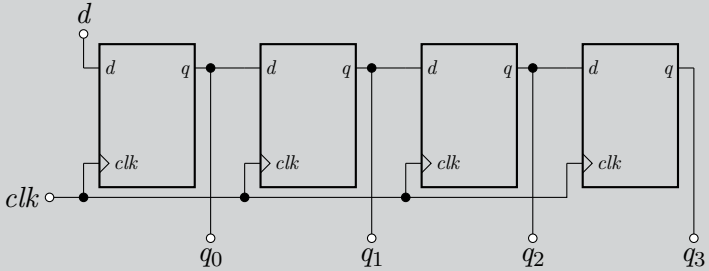


Combinatorial and Sequential Circuits

Combinatorial Circuit



Sequential Circuit (e.g. Shift Register)



Synthesizable System Verilog

We actually only need two `always` statements:

Combinational Processes

```
always_comb begin
    if ((c==1) & (d==1))
        y = b;
    else
        y = a;
end
```

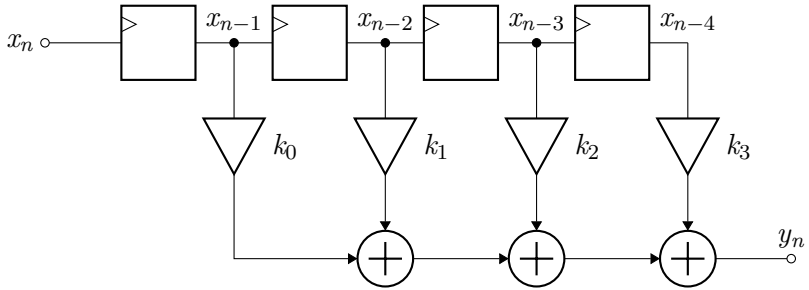
- Uses `always_comb` statement
- Blocking assignment (=)
- No feedback loops allowed (should be avoided anyway)

Clocked Processes

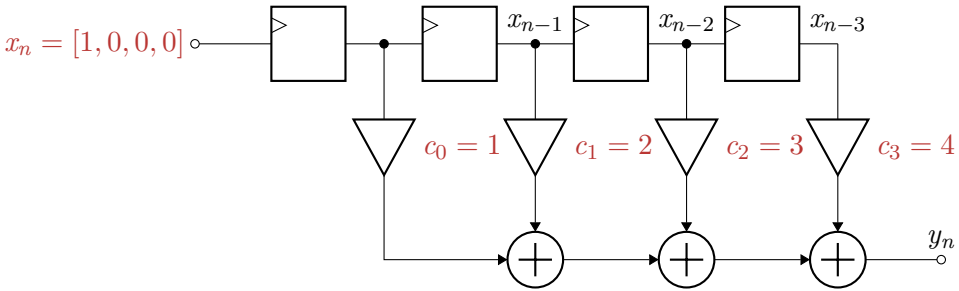
```
always_ff @(posedge clk)
begin
    r <= {d, r[3:1]};
    q3 <= r[0];
end

```

- Uses `always_ff` statement
- Non-Blocking assignment (<=)
- Always generates Flip-Flops

Example: FIR Filter



Example: FIR Filter

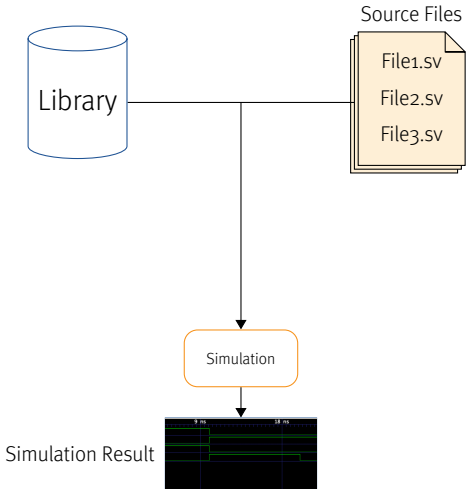


Example: FIR Filter

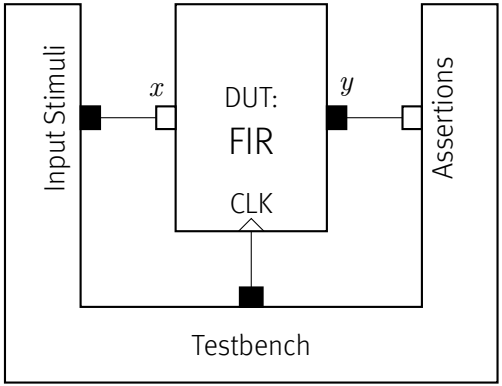

```
module fir (
    input logic clk,      // Clock Signal
    input logic rst,      // Reset Signal
    input logic [7:0] x, // Input Signal
    output logic [7:0] y // Output Signal
);

    // Registers and Coefficients:
    logic [7:0] r [3:0];
    logic [7:0] c [3:0];

    initial c[0] = 8'h01;
    initial c[1] = 8'h02;
    initial c[2] = 8'h03;
    initial c[3] = 8'h04;
```

```
    always_ff @( posedge clk ) begin
        if (rst) begin
            // Reset everything to zero
            r[0] <= 8'h00;
            r[1] <= 8'h00;
            r[2] <= 8'h00;
            r[3] <= 8'h00;
        end else begin
            r[0] <= x;
            r[1] <= r[0];
            r[2] <= r[1];
            r[3] <= r[2];
        end
    end

    always_comb begin
        y = r[0] * c[0]
          + r[1] * c[1]
          + r[2] * c[2]
          + r[3] * c[3];
    end
endmodule
```

Verilog Simulations-Flow



Testbenches

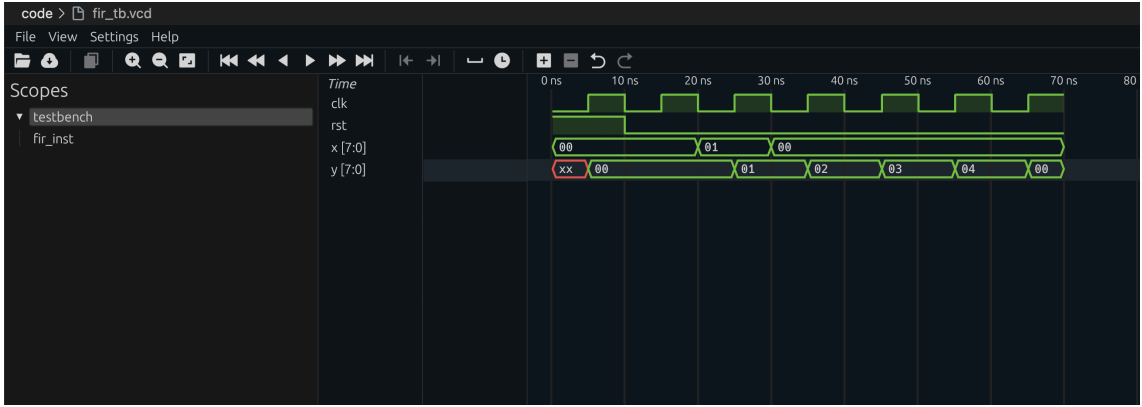


A testbench generates stimuli for a *Design Under Test* (DUT) to simulate and verify its behavior.

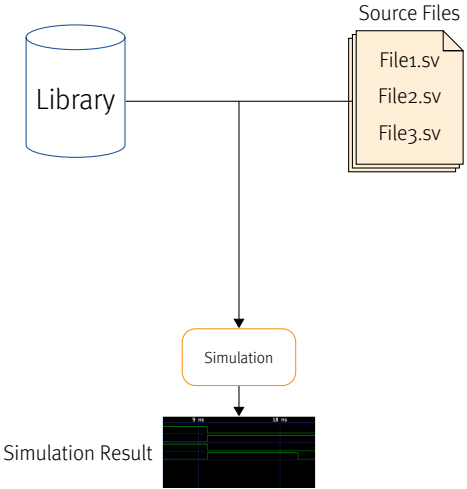
```
initial begin
    $dumpfile("fir_tb.vcd");
    $dumpvars(0, testbench);
    rst = 1;
    x = 16'h0000;
    #10 rst = 0; // Release reset after 10 ns

    // Apply test inputs
    x = 8'h00; #10;
    x = 8'h01; #10;
    assert (y == 8'h01);
    x = 8'h00; #10;
    assert (y == 8'h02);
    x = 8'h00; #10;
    assert (y == 8'h03);
    x = 8'h00; #10;
    assert (y == 8'h04);
    x = 8'h00; #10;
    assert (y == 8'h00);
end
```

Simulation Results

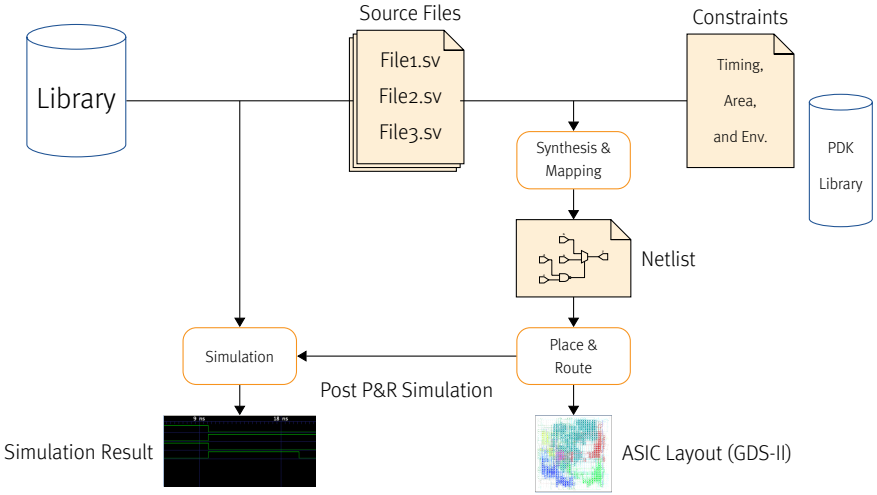
Signals can be viewed in a waveform viewer, e.g., Surfer or GTKWave:



Generic Development Flow



Generic Development Flow



Open Source Synthesis Flow and Tiny Tapeout

Open Source Synthesis Flow:

- Open Source PDKs: Google Skywater PDK, IHP130, ...
- Open Source Simulation and Synthesis Tools: iVerilog, Yosys, Nextpnr, OpenROAD, Surfer, GTKWave, OSS-CAD-Suite...

Open Source Synthesis Flow and Tiny Tapeout

Open Source Synthesis Flow:

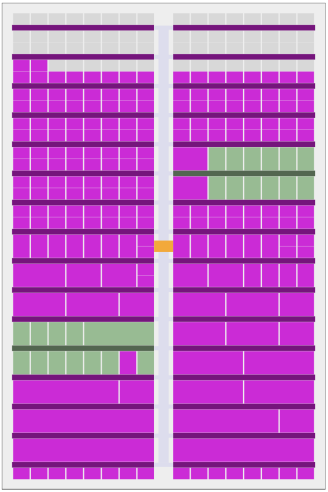
- Open Source PDKs: Google Skywater PDK, IHP130, ...
- Open Source Simulation and Synthesis Tools: iVerilog, Yosys, Nextpnr, OpenROAD, Surfer, GTKWave, OSS-CAD-Suite...

Tiny Tapeout Initiative:

- Project to get startet with Open Source ASIC Design
- Low-Cost ($\sim$ 150$)
- Your design must be open source
- $\sim$ 100 projects share the real-estate on the same chip

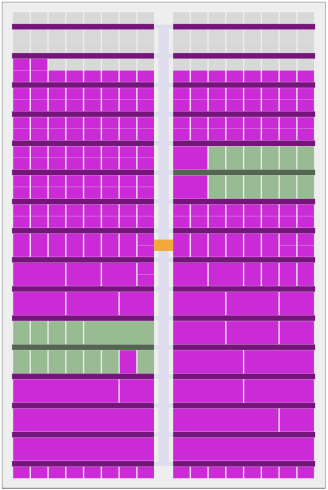
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit



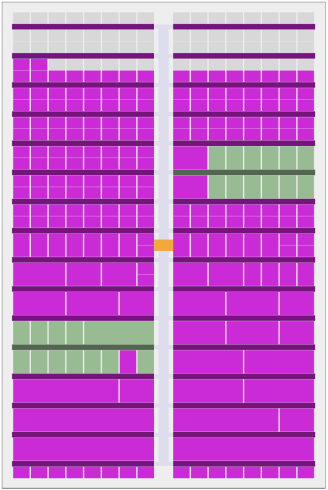
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles



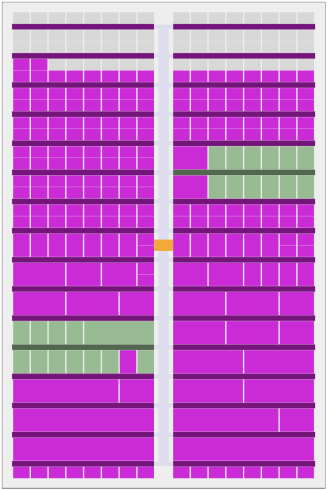
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells



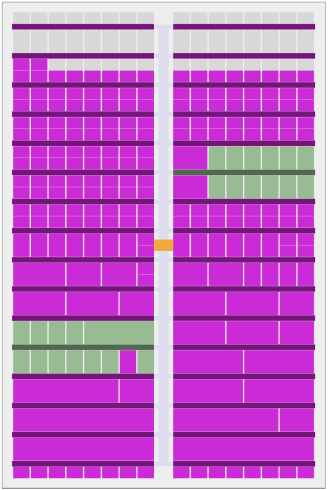
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:



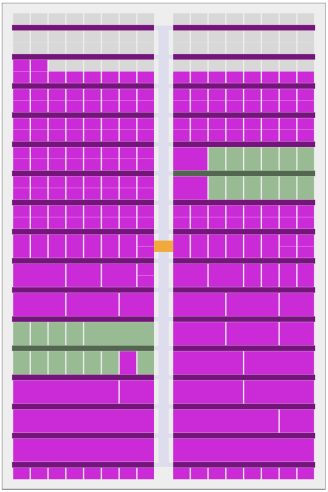
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing



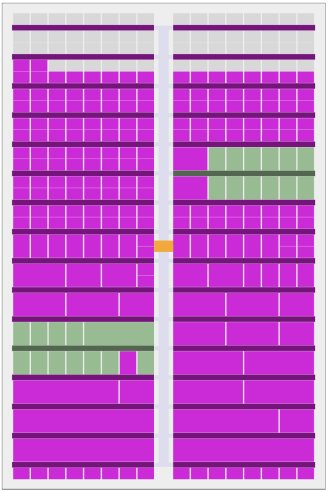
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing
- Designs can be selected by DIP switches or software



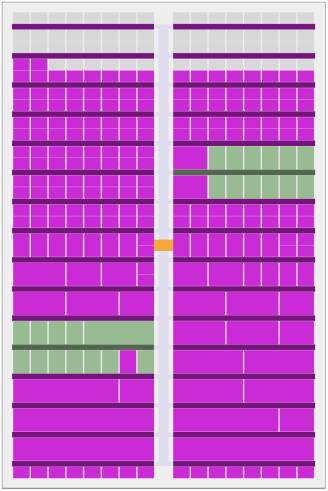
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing
- Designs can be selected by DIP switches or software
- Green tiles show analog projects



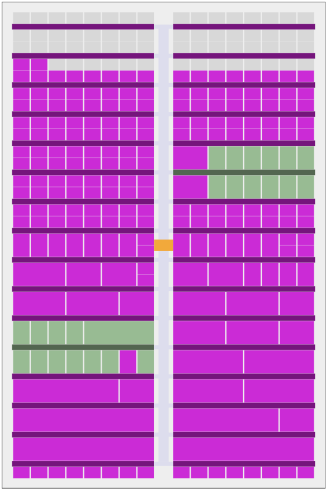
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing
- Designs can be selected by DIP switches or software
- Green tiles show analog projects
- Example shows TT06 where my students submitted a RISC-V



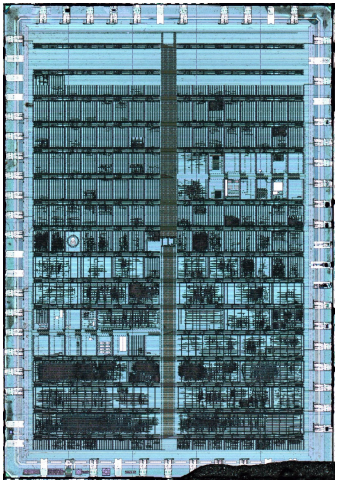
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing
- Designs can be selected by DIP switches or software
- Green tiles show analog projects
- Example shows TT06 where my students submitted a RISC-V



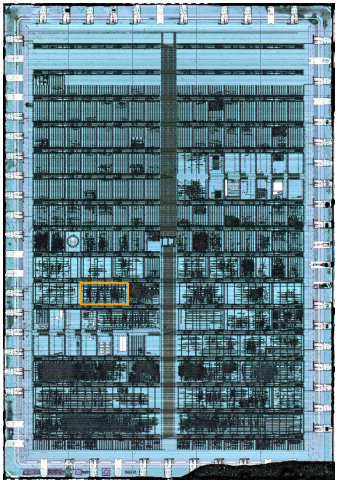
Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing
- Designs can be selected by DIP switches or software
- Green tiles show analog projects
- Example shows TT06 where my students submitted a RISC-V

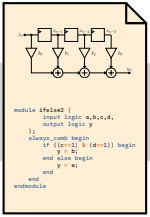


Tiny Tapeout Chip

- A »Tile« is the minimum bookable unit
- A project can consist of 1, 2, 4 or 8 tiles
- Each tile can host around 1000 standard cells
- Number of IOs:
  - 8 In
  - 8 Out
  - 8 In/Out
- Large Multiplexer (MUX) for routing
- Designs can be selected by DIP switches or software
- Green tiles show analog projects
- Example shows TT06 where my students submitted a RISC-V

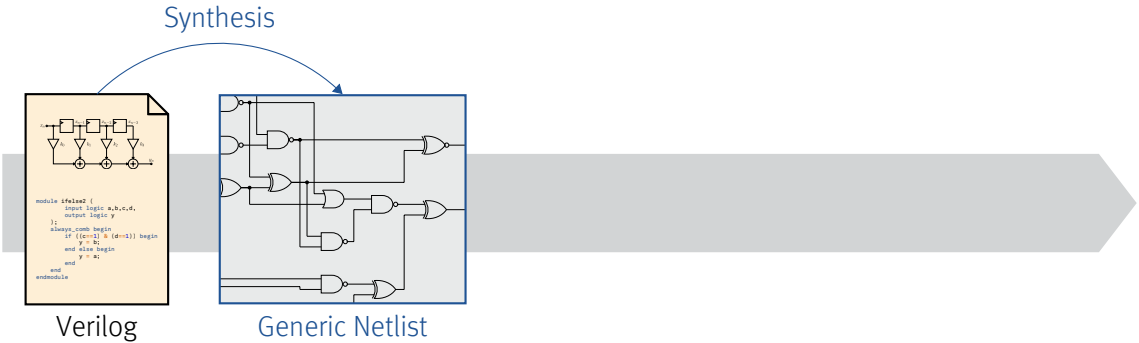


Example Synthesis

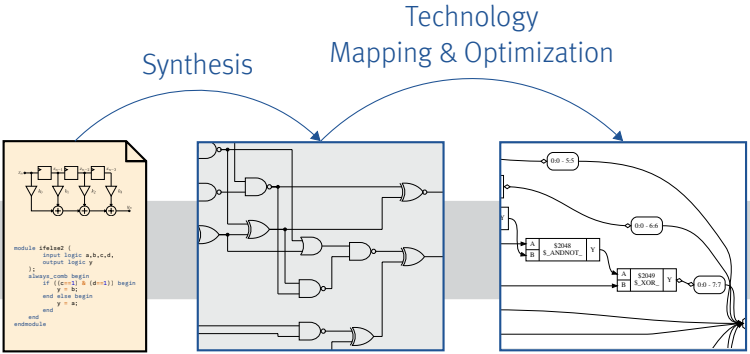


Verilog

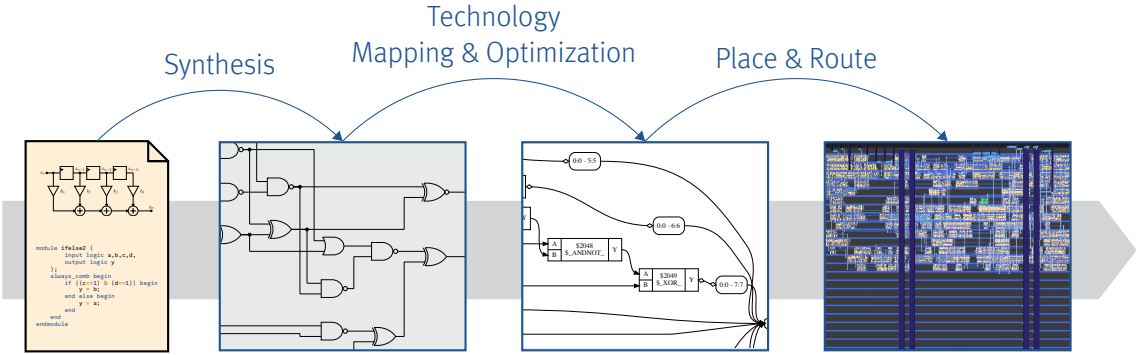
Example Synthesis



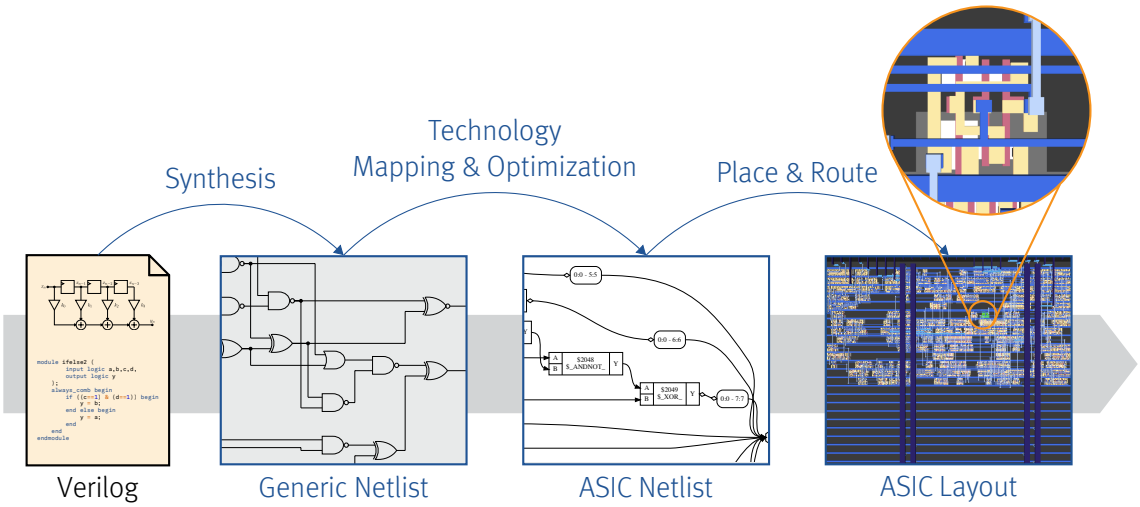
Example Synthesis



Example Synthesis



Example Synthesis



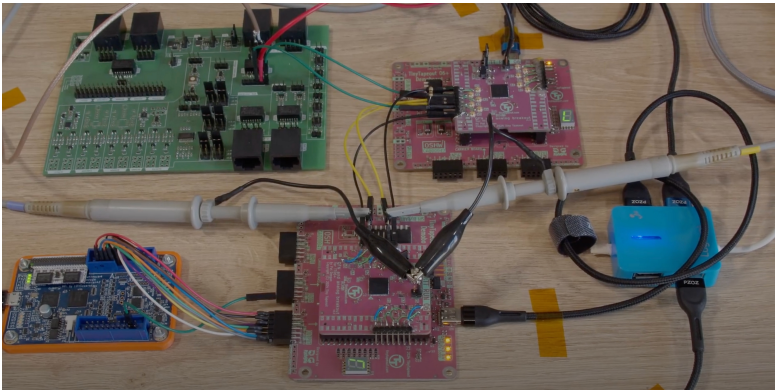
Synthesis Results

Utilization: 15.26%, 269 total cells, wire Length: 3909 µm, GDS 3D Viewer

Type	Standard Cells	Number
AND	and2 and3 and4 a21boi	46
NOR	nor2 xnor2	46
Combo Logic	a210 o21ai and2b a210i a310 o22a nor3b and3b a2110 a320 a220 o21a a2bb20	42
Misc	conb dlygate4sd3	36
Buffer	buf clkbuf	34
Flip Flops	dfxtp	30
OR	or2 xor2	18
NAND	nand2 nand2b nand3	15
Inverter	inv	2
Multiplexer	mux2	2

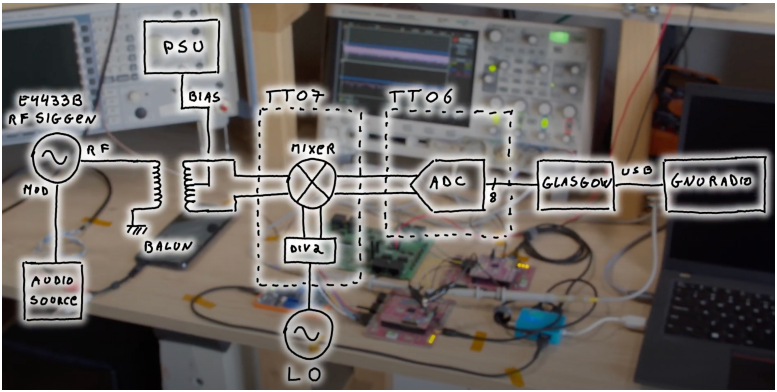
First »SDR« on Tiny Tapeout

- Sylvain Munaut created first SDR with Tiny Tapeout chips
- 8 bit ADC from Carsten Wulff in TT06
- Gilbert cell RF mixer from Kolos Koblász in TT07)
- Rest is done with GNU Radio
- Watch on YouTube



First »SDR« on Tiny Tapeout

- Sylvain Munaut created first SDR with Tiny Tapeout chips
- 8 bit ADC from Carsten Wulff in TT06
- Gilbert cell RF mixer from Kolos Koblász in TT07)
- Rest is done with GNU Radio
- Watch on YouTube



Conclusion

- Open-Source EDA Software and Open-Source PDFKs democratize chipdesign

Conclusion

- Open-Source EDA Software and Open-Source PDFKs democratize chipdesign
- Tiny Tapeout, maybe not enough for a full SDR design, e.g. the number of IOs is limited

Conclusion

- Open-Source EDA Software and Open-Source PDFKs democratize chipdesign
- Tiny Tapeout, maybe not enough for a full SDR design, e.g. the number of IOs is limited
- Nevertheless, Tiny Tapeout is a great way to get started with ASIC Design

Conclusion

- Open-Source EDA Software and Open-Source PDFKs democratize chipdesign
- Tiny Tapeout, maybe not enough for a full SDR design, e.g. the number of IOs is limited
- Nevertheless, Tiny Tapeout is a great way to get started with ASIC Design
- Many HAMs could join forces to do a MWP (e.g. via `https://chipfoundry.io`, or IHP)

Conclusion

- Open-Source EDA Software and Open-Source PDFKs democratize chipdesign
- Tiny Tapeout, maybe not enough for a full SDR design, e.g. the number of IOs is limited
- Nevertheless, Tiny Tapeout is a great way to get started with ASIC Design
- Many HAMs could join forces to do a MWP (e.g. via `https://chipfoundry.io`, or IHP)

Questions ?

Prof. Dr.-Ing. Matthias Jung (DL9MJ)
dl9mj@darc.de
go.uniwue.de/ce