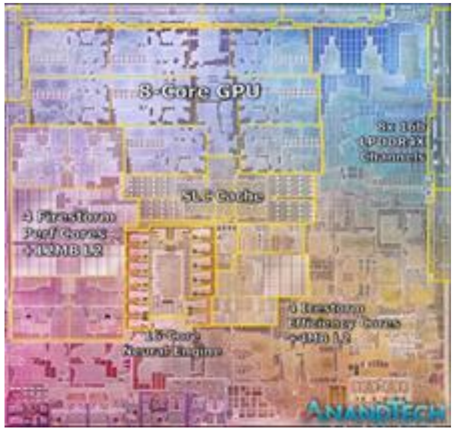
mflowgen: A Modular Flow Generator and Ecosystem for Community-Driven Physical Design

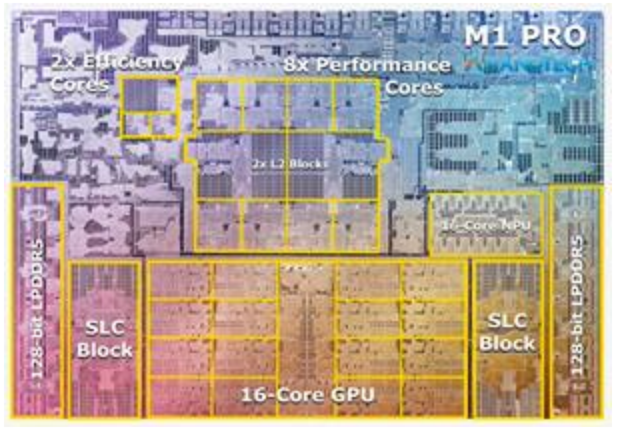
Christopher Torng

Stanford University

Complex systems are not easy to build



Apple M1 (2020)
***Transistor Count**: 16 billion*

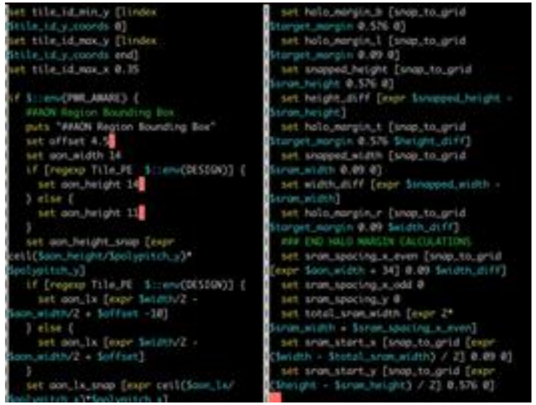


Apple M1 Pro (2021)
***Transistor Count**: 34 billion*

The required expertise in **physical design** to work on such systems is high

How do we reduce the **design effort** of building complex systems?

Monolithic Tcl Scripts

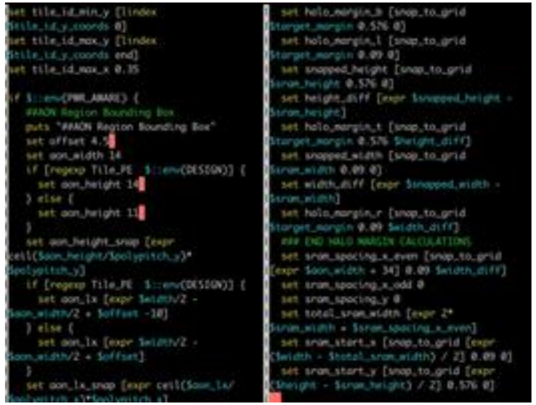


```
set tile_id_min_y [lindex
$tile_id_y_coords 0]
set tile_id_max_y [lindex
$tile_id_y_coords end]
set tile_id_max_x 0.35

if $::env(PWR_AWARE) {
  ##AON Region Bounding Box
  puts "##AON Region Bounding Box"
  set offset 4.5
  set aon_width 14
  if [regexp Tile_PE  $::env(DESIGN)] {
    set aon_height 14
  } else {
    set aon_height 11
  }
  set aon_height_snap [expr
ceil($aon_height/$polypitch_y)*
$polypitch_y]
  if [regexp Tile_PE  $::env(DESIGN)] {
    set aon_lx [expr $width/2 -
$aon_width/2 + $offset -10]
  } else {
    set aon_lx [expr $width/2 -
$aon_width/2 + $offset]
  }
  set aon_lx_snap [expr ceil($aon_lx/
$polypitch_x)*$polypitch_x]
```

```
  set halo_margin_b [snap_to_grid
$target_margin 0.576 0]
  set halo_margin_l [snap_to_grid
$target_margin 0.09 0]
  set snapped_height [snap_to_grid
$sram_height 0.576 0]
  set height_diff [expr $snapped_height -
$sram_height]
  set halo_margin_t [snap_to_grid
$target_margin 0.576 $height_diff]
  set snapped_width [snap_to_grid
$sram_width 0.09 0]
  set width_diff [expr $snapped_width -
$sram_width]
  set halo_margin_r [snap_to_grid
$target_margin 0.09 $width_diff]
  ### END HALO MARGIN CALCULATIONS
  set sram_spacing_x_even [snap_to_grid
[expr $aon_width + 34] 0.09 $width_diff]
  set sram_spacing_x_odd 0
  set sram_spacing_y 0
  set total_sram_width [expr 2*
$sram_width + $sram_spacing_x_even]
  set sram_start_x [snap_to_grid [expr
($width - $total_sram_width) / 2] 0.09 0]
  set sram_start_y [snap_to_grid [expr
($height - $sram_height) / 2] 0.576 0]
```

How do we reduce the **design effort** of building complex systems?

Monolithic Tcl Scripts



```
set tile_id_min_y [lindex
$tile_id_y_coords 0]
set tile_id_max_y [lindex
$tile_id_y_coords end]
set tile_id_max_x 0.35

if $::env(PWR_AWARE) {
  ##AON Region Bounding Box
  puts "##AON Region Bounding Box"
  set offset 4.5
  set aon_width 14
  if [regexp Tile_PE  $::env(DESIGN)] {
    set aon_height 14
  } else {
    set aon_height 11
  }
  set aon_height_snap [expr
ceil($aon_height/$polypitch_y)*
$polypitch_y]
  if [regexp Tile_PE  $::env(DESIGN)] {
    set aon_lx [expr $width/2 -
$aon_width/2 + $offset -10]
  } else {
    set aon_lx [expr $width/2 -
$aon_width/2 + $offset]
  }
  set aon_lx_snap [expr ceil($aon_lx/
$polypitch_x)*$polypitch_x]
```

```
  set halo_margin_b [snap_to_grid
$target_margin 0.576 0]
  set halo_margin_l [snap_to_grid
$target_margin 0.09 0]
  set snapped_height [snap_to_grid
$sram_height 0.576 0]
  set height_diff [expr $snapped_height -
$sram_height]
  set halo_margin_t [snap_to_grid
$target_margin 0.576 $height_diff]
  set snapped_width [snap_to_grid
$sram_width 0.09 0]
  set width_diff [expr $snapped_width -
$sram_width]
  set halo_margin_r [snap_to_grid
$target_margin 0.09 $width_diff]
  ### END HALO MARGIN CALCULATIONS
  set sram_spacing_x_even [snap_to_grid
[expr $aon_width + 34] 0.09 $width_diff]
  set sram_spacing_x_odd 0
  set sram_spacing_y 0
  set total_sram_width [expr 2*
$sram_width + $sram_spacing_x_even]
  set sram_start_x [snap_to_grid [expr
($width - $total_sram_width) / 2] 0.09 0]
  set sram_start_y [snap_to_grid [expr
($height - $sram_height) / 2] 0.576 0]
```

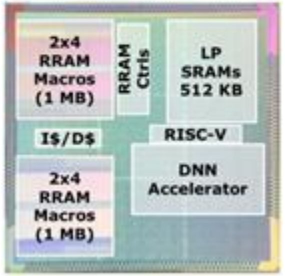
Opportunities for reuse are lost

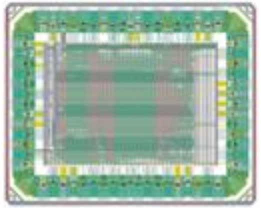
- Customizing for a design
- Customizing for a technology

How can we enable **community-driven physical design**?

with a university as a testing ground?

TSMC 16nm

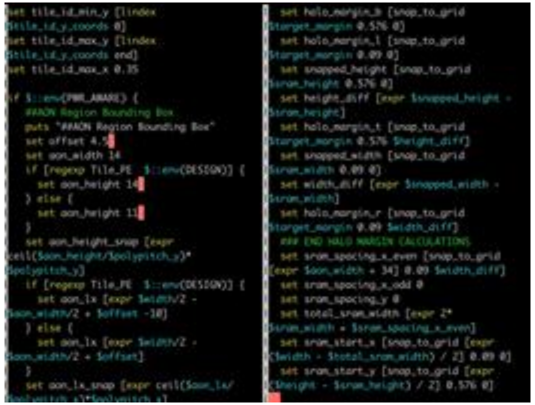




TSMC 40nm **TSMC 28nm**

How do we reduce the **design effort** of building complex systems?

Monolithic Tcl Scripts



```
set tile_id_min_y [lindex
$tile_id_y_coords 0]
set tile_id_max_y [lindex
$tile_id_y_coords end]
set tile_id_max_x 0.35

if $::env(PWR_AWARE) {
  ##AON Region Bounding Box
  puts "##AON Region Bounding Box"
  set offset 4.5
  set aon_width 14
  if [regexp Tile_PE  $::env(DESIGN)] {
    set aon_height 14
  } else {
    set aon_height 11
  }
  set aon_height_snap [expr
ceil($aon_height/$polypitch_y)*
$polypitch_y]
  if [regexp Tile_PE  $::env(DESIGN)] {
    set aon_lx [expr $width/2 -
$aon_width/2 + $offset -10]
  } else {
    set aon_lx [expr $width/2 -
$aon_width/2 + $offset]
  }
  set aon_lx_snap [expr ceil($aon_lx/
$polypitch_x)*$polypitch_x]
```

```
  set halo_margin_b [snap_to_grid
$target_margin 0.576 0]
  set halo_margin_l [snap_to_grid
$target_margin 0.09 0]
  set snapped_height [snap_to_grid
$sram_height 0.576 0]
  set height_diff [expr $snapped_height -
$sram_height]
  set halo_margin_t [snap_to_grid
$target_margin 0.576 $height_diff]
  set snapped_width [snap_to_grid
$sram_width 0.09 0]
  set width_diff [expr $snapped_width -
$sram_width]
  set halo_margin_r [snap_to_grid
$target_margin 0.09 $width_diff]
  ### END HALO MARGIN CALCULATIONS
  set sram_spacing_x_even [snap_to_grid
[expr $aon_width + 34] 0.09 $width_diff]
  set sram_spacing_x_odd 0
  set sram_spacing_y 0
  set total_sram_width [expr 2*
$sram_width + $sram_spacing_x_even]
  set sram_start_x [snap_to_grid [expr
($width - $total_sram_width) / 2] 0.09 0]
  set sram_start_y [snap_to_grid [expr
($height - $sram_height) / 2] 0.576 0]
```

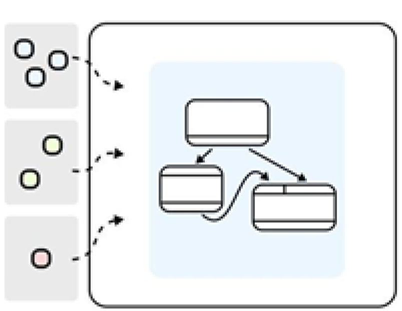
Opportunities for reuse are lost

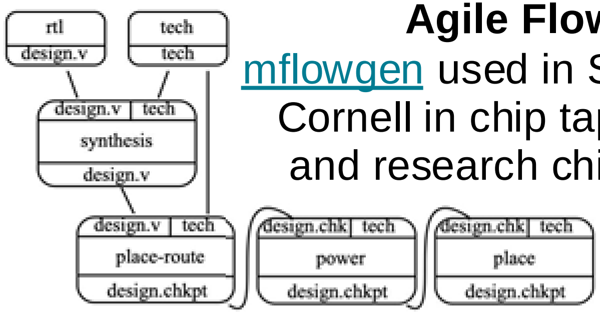
- Customizing for a design
- Customizing for a technology

How can we enable **community-driven physical design**?

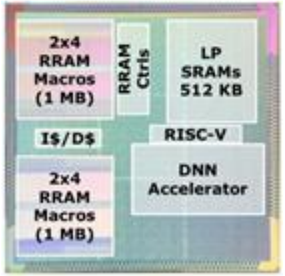
with a university as a testing ground?

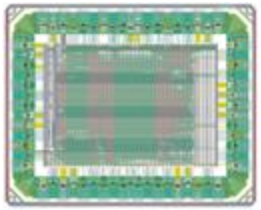
TSMC 16nm





Agile Flow Tool:
mflowgen used in Stanford / USC / Cornell in chip tapeout courses and research chip prototyping

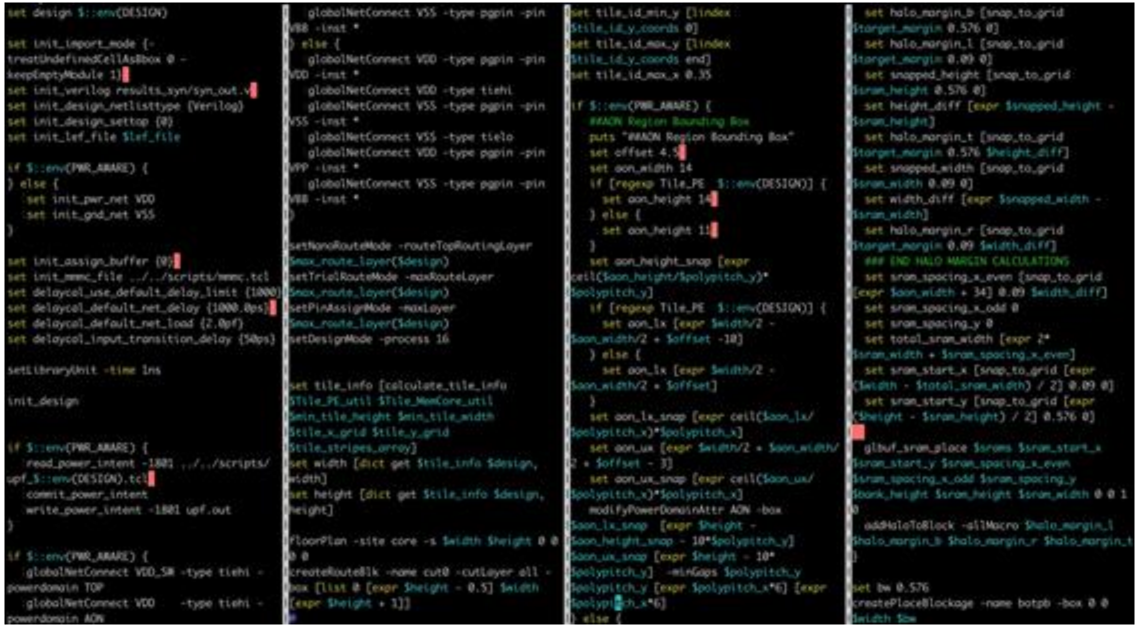




TSMC 40nm **TSMC 28nm**

Reuse as much of previous systems as we can

… but this is very challenging



```
set design $::env(DESIGN)

set init_import_mode {-
treatUndefinedCellAsBbox 0 -
keepEmptyModule 1}
set init_verilog results_syn/syn_out.v
set init_design_netlisttype {Verilog}
set init_design_settop {0}
set init_lef_file $lef_file

if $::env(PWR_AWARE) {
} else {
  set init_pwr_net VDD
  set init_gnd_net VSS
}

set init_assign_buffer {0}
set init_mmmc_file ../../scripts/mmmc.tcl
set delaycal_use_default_delay_limit {1000}
set delaycal_default_net_delay {1000.0ps}
set delaycal_default_net_load {2.0pf}
set delaycal_input_transition_delay {50ps}

setLibraryUnit -time 1ns

init_design


if $::env(PWR_AWARE) {
   read_power_intent -1801 ../../scripts/
upf_$::env(DESIGN).tcl
   commit_power_intent
   write_power_intent -1801 upf.out
}

if $::env(PWR_AWARE) {
   globalNetConnect VDD_SW -type tiehi -
powerdomain TOP
   globalNetConnect VDD    -type tiehi -
powerdomain AON
```

```
   globalNetConnect VSS -type pgpin -pin
VBB -inst *
} else {
   globalNetConnect VDD -type pgpin -pin
VDD -inst *
   globalNetConnect VDD -type tiehi
   globalNetConnect VSS -type pgpin -pin
VSS -inst *
   globalNetConnect VSS -type tielo
   globalNetConnect VDD -type pgpin -pin
VPP -inst *
   globalNetConnect VSS -type pgpin -pin
VBB -inst *
}

setNanoRouteMode -routeTopRoutingLayer
$max_route_layer($design)
setTrialRouteMode -maxRouteLayer
$max_route_layer($design)
setPinAssignMode -maxLayer
$max_route_layer($design)
setDesignMode -process 16


set tile_info [calculate_tile_info
$Tile_PE_util $Tile_MemCore_util
$min_tile_height $min_tile_width
$tile_x_grid $tile_y_grid
$tile_stripes_array]
set width [dict get $tile_info $design,
width]
set height [dict get $tile_info $design,
height]

floorPlan -site core -s $width $height 0 0
0 0
createRouteBlk -name cut0 -cutLayer all -
box [list 0 [expr $height - 0.5] $width
[expr $height + 1]]
```

```
set tile_id_min_y [lindex
$tile_id_y_coords 0]
set tile_id_max_y [lindex
$tile_id_y_coords end]
set tile_id_max_x 0.35

if $::env(PWR_AWARE) {
   ##AON Region Bounding Box
   puts "##AON Region Bounding Box"
   set offset 4.5
   set aon_width 14
   if [regexp Tile_PE  $::env(DESIGN)] {
     set aon_height 14
   } else {
     set aon_height 11
   }
   set aon_height_snap [expr
ceil($aon_height/$polypitch_y)*
$polypitch_y]
   if [regexp Tile_PE  $::env(DESIGN)] {
     set aon_lx [expr $width/2 -
$aon_width/2 + $offset -10]
   } else {
     set aon_lx [expr $width/2 -
$aon_width/2 + $offset]
   }
   set aon_lx_snap [expr ceil($aon_lx/
$polypitch_x)*$polypitch_x]
   set aon_ux [expr $width/2 + $aon_width/
2 + $offset - 3]
   set aon_ux_snap [expr ceil($aon_ux/
$polypitch_x)*$polypitch_x]
   modifyPowerDomainAttr AON -box
$aon_lx_snap  [expr $height -
$aon_height_snap - 10*$polypitch_y]
$aon_ux_snap [expr $height - 10*
$polypitch_y]  -minGaps $polypitch_y
$polypitch_y [expr $polypitch_x*6] [expr
$polypitch_x*6]
} else {
```

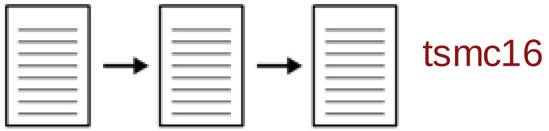
```
  set halo_margin_b [snap_to_grid
$target_margin 0.576 0]
  set halo_margin_l [snap_to_grid
$target_margin 0.09 0]
  set snapped_height [snap_to_grid
$sram_height 0.576 0]
  set height_diff [expr $snapped_height -
$sram_height]
  set halo_margin_t [snap_to_grid
$target_margin 0.576 $height_diff]
  set snapped_width [snap_to_grid
$sram_width 0.09 0]
  set width_diff [expr $snapped_width -
$sram_width]
  set halo_margin_r [snap_to_grid
$target_margin 0.09 $width_diff]
  ### END HALO MARGIN CALCULATIONS
  set sram_spacing_x_even [snap_to_grid
[expr $aon_width + 34] 0.09 $width_diff]
  set sram_spacing_x_odd 0
  set sram_spacing_y 0
  set total_sram_width [expr 2*
$sram_width + $sram_spacing_x_even]
  set sram_start_x [snap_to_grid [expr
($width - $total_sram_width) / 2] 0.09 0]
  set sram_start_y [snap_to_grid [expr
($height - $sram_height) / 2] 0.576 0]

  glbuf_sram_place $srams $sram_start_x
$sram_start_y $sram_spacing_x_even
$sram_spacing_x_odd $sram_spacing_y
$bank_height $sram_height $sram_width 0 0 1
0
  addHaloToBlock -allMacro $halo_margin_l
$halo_margin_b $halo_margin_r $halo_margin_t
}

set bw 0.576
createPlaceBlockage -name botpb -box 0 0
$width $bw
```

Long TCL files that accomplish many things

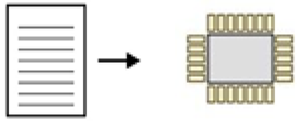
Where is the reusable snippet of code?

If I add commands, will it break code elsewhere in this file?

Code customization prevents future reuse



1. Scripts tied to a particular **technology** (process node, library vendors, pdk)



2. Scripts tied to a particular **design** (custom power strategy, physical tiling, abutment)

Physical design flows look similar yet different

Project #1

```
synthesis
  ...

floorplan
  ...

place
  ...

route
  ...
```

Technology: 16nm

Design: Crypto Accel

Project #2

```
synthesis
  ...

floorplan
  ...

place
  ...

route
  ...
```

Technology: 12nm

Design: Video Decoder

Monolithic Tcl flows

... mostly the same

... but customized differently

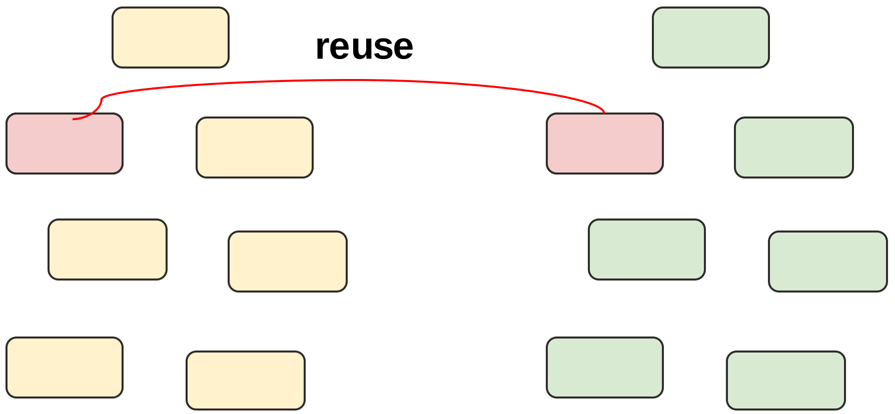
... with little reuse

Is there a **better approach** to reusing physical design code?

Key Idea: Embrace Modularity

Project #1

Project #2

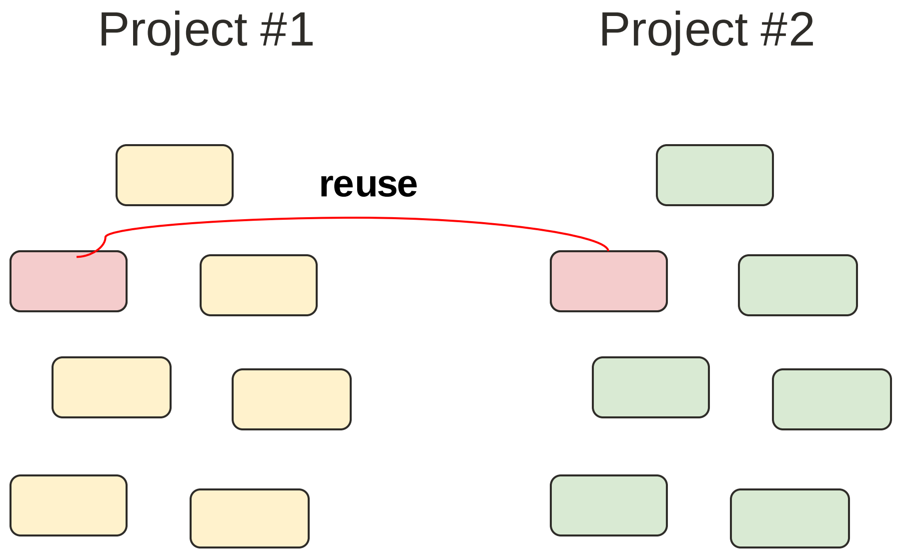


Technology: 16nm
Design: Crypto Accel

Technology: 12nm
Design: Video Decoder

Hypothesis #1: There is common code we can directly reuse despite custom code

Key Idea: Embrace Modularity



Technology: 16nm
Design: Crypto Accel

Technology: 12nm
Design: Video Decoder

Hypothesis #1: There is common code we can directly reuse despite custom code

Hypothesis #2: We can construct an *overwhelming majority* of a new project from such pieces

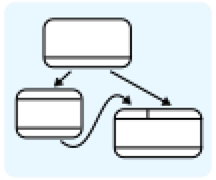
Agile Flow Tools

System Goals

Design principles that enable and maximize reuse

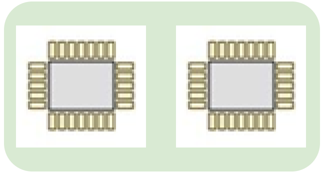
Modular Flow Generators

Abstractions and DSL to build and generate modular flows



Community

Features to support community-driven physical design



Agile Flow Tool: System Goals

System Goal	Observation	Requirement
Code Reuse - "To meaningfully reduce design effort, we require *significant* code reuse"		

Agile Flow Tool: System Goals

System Goal	Observation	Requirement
Code Reuse - "To meaningfully reduce design effort, we require *significant* code reuse"	Need to reuse ***an extremely high degree*** of the physical design flow (90%+)	

Agile Flow Tool: System Goals

System Goal

Code Reuse - "To meaningfully reduce design effort, we require *significant* code reuse"

Observation

Need to reuse ***an extremely high degree*** of the physical design flow (90%+)

Requirement

Capture both **coarse-grain reuse** and **fine-grain reuse**

Support mechanisms to **tweak reusable code** in small ways

Agile Flow Tool: System Goals

System Goal	Observation	Requirement
Code Reuse - "To meaningfully reduce design effort, we require *significant* code reuse"	Need to reuse ***an extremely high degree*** of the physical design flow (90%+)	Capture both **coarse-grain reuse** and **fine-grain reuse** Support mechanisms to **tweak reusable code** in small ways
Code Reuse - "Composition must support code from different designs and technologies"	Existing flows will unavoidably be customized for specific designs and technologies	Mechanism to check for **composability** of flow scripts and code fragments
		Needs a **static code analysis** approach because flow scripts are distributed across tools, **not in memory at the same time**

Agile Flow Tool: System Goals

System Goal	Observation	Requirement
Code Reuse - "To meaningfully reduce design effort, we require *significant* code reuse"	Need to reuse ***an extremely high degree*** of the physical design flow (90%+)	Capture both **coarse-grain reuse** and **fine-grain reuse** Support mechanisms to **tweak reusable code** in small ways
Code Reuse - "Composition must support code from different designs and technologies"	Existing flows will unavoidably be customized for specific designs and technologies	Mechanism to check for **composability** of flow scripts and code fragments
Rapid Feedback - "Feedback on inconsistent composition must be both rapid and early"	Inconsistent composition can easily break any newly composed flow Dynamic checks are slow because **physical design tools run for ~days**	Needs a **static code analysis** approach because flow scripts are distributed across tools, **not in memory at the same time**

Stanford University

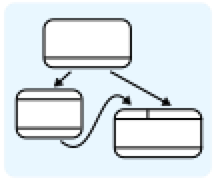
Agile Flow Tools

System Goals

Design principles that enable and maximize reuse

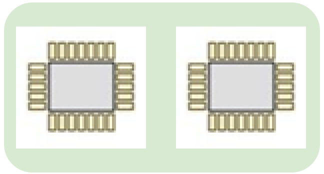
Modular Flow Generators

Abstractions and DSL to build and generate modular flows



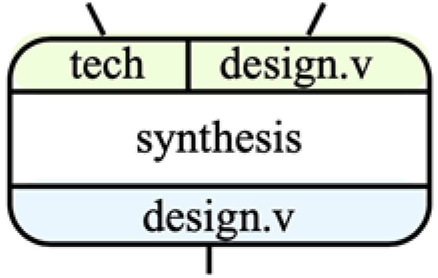
Community

Features to support community-driven physical design



Modular Nodes

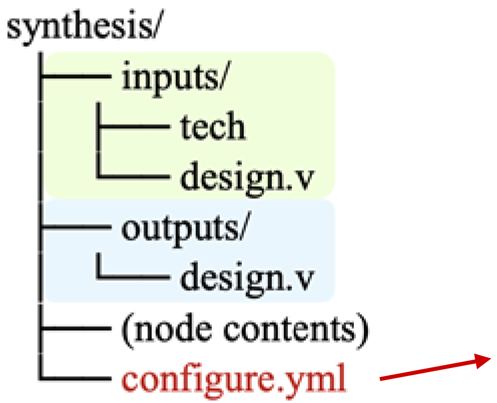
Graph View



Function signature with file-based inputs and outputs

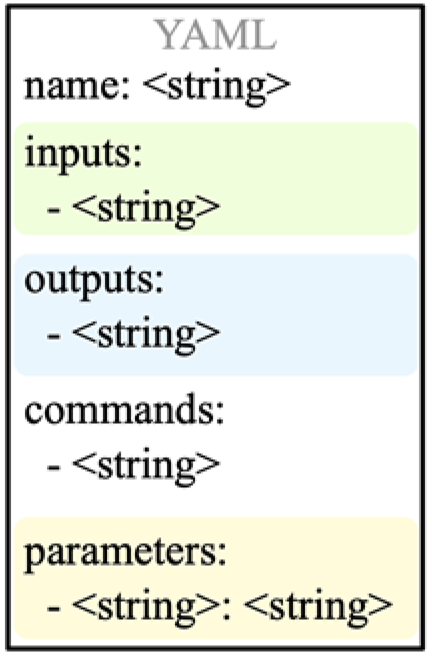
Recall Goal #1: Capture both fine-grain and coarse-grain reuse

File System View



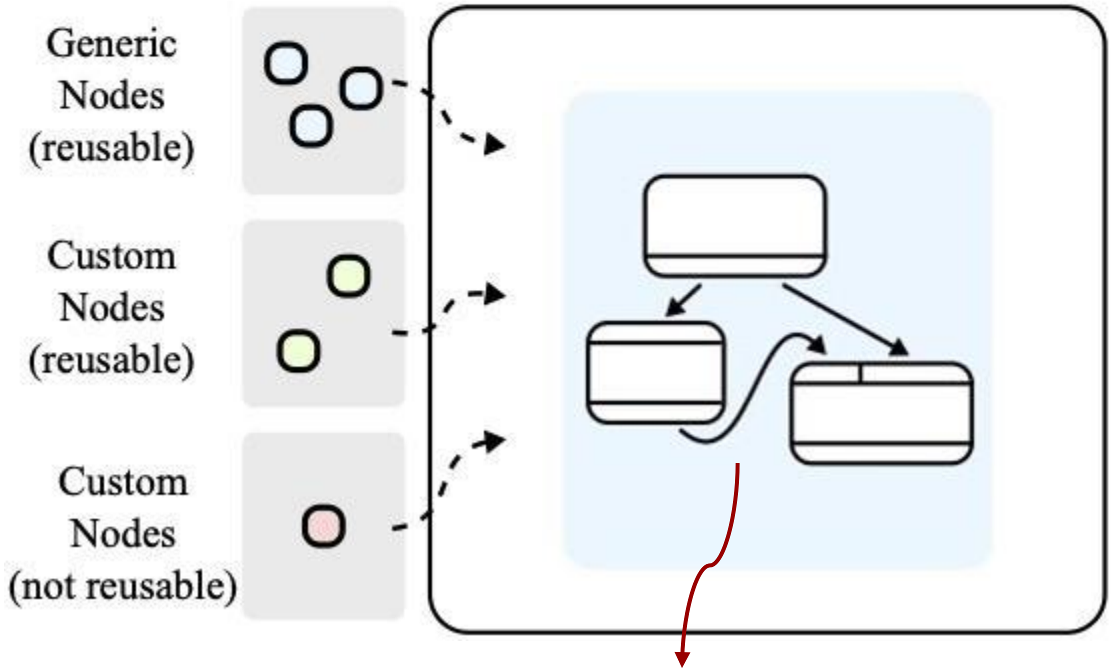
Nodes are islands, **decoupled** from the source of their inputs

Configuration Schema



```
YAML
name: <string>
inputs:
  - <string>
outputs:
  - <string>
commands:
  - <string>
parameters:
  - <string>: <string>
```

Modular Flow Generator



Python-Embedded Graph-Building DSL

```
g = Graph()

g.add_node( .. )
g.connect( .. )

g.update_params( .. )
```

Recall Goals #1 and #2: DSL allows flexibility to reuse 90%+ of graph

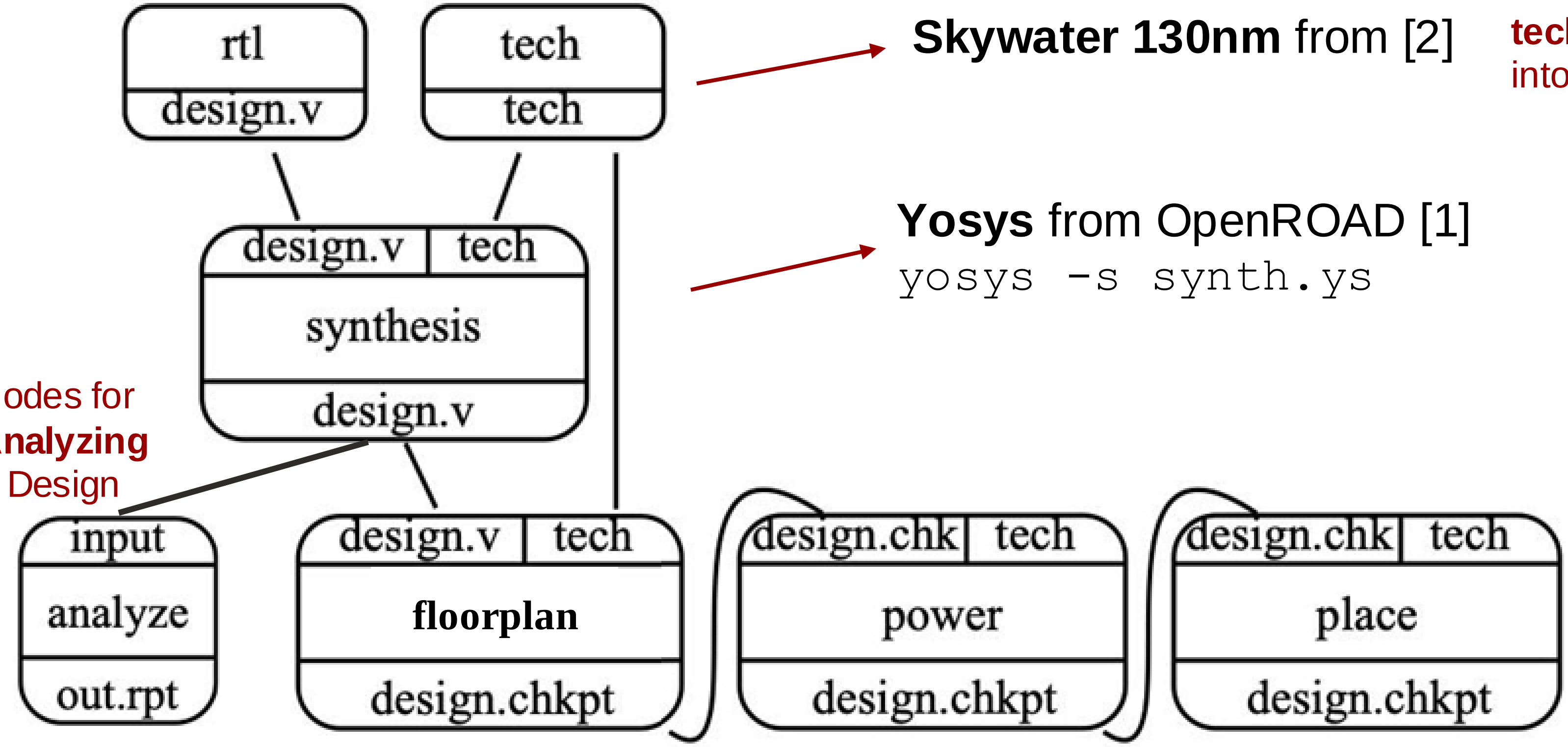
Edges: automatically generates code that moves (links) files to next island

Example Teaching Graph

[1] https://theopenroadproject.org
[2] https://github.com/google/skywater-pdk

Other nodes are
Static Vendor Packages

Abstract the
technology files
into one node



Skywater 130nm from [2]

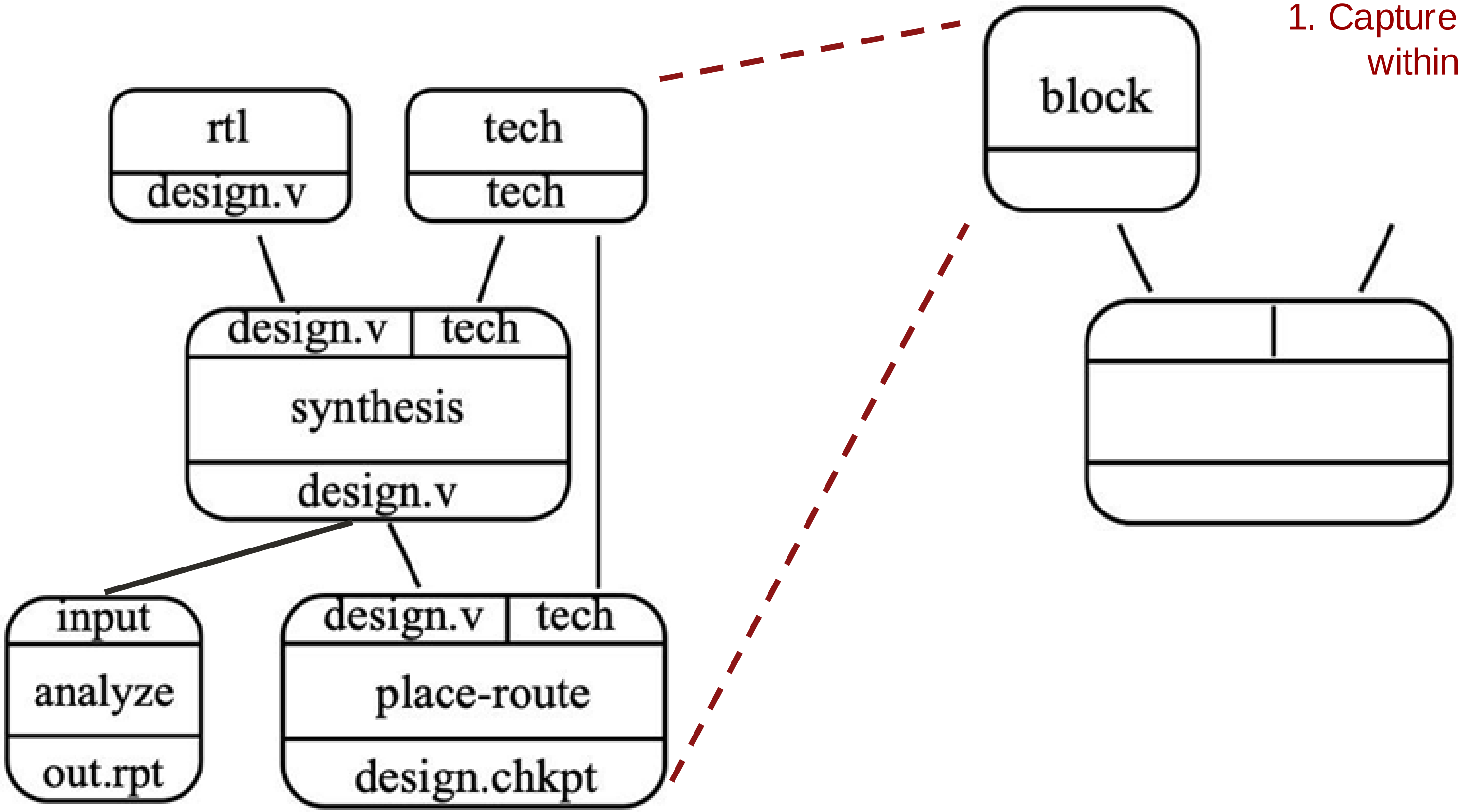
Yosys from OpenROAD [1]

```
yosys -s synth.ys
```

Nodes for
Analyzing
a Design

Nodes for
Transforming
a Design

Naturally capture hierarchical graphs



1. Capture a **subgraph** within a node

2. Connect to other nodes to create a **hierarchical flow**

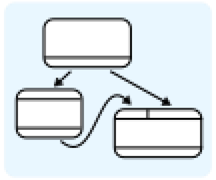
Agile Flow Tools

System Goals

Design principles that enable and maximize reuse

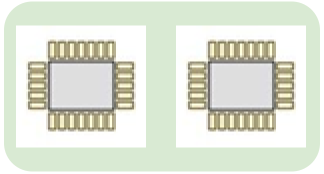
Modular Flow Generators

Abstractions and DSL to build and generate modular flows



Community

Features to support community-driven physical design



mflowgen Demo - Greatest common divisor

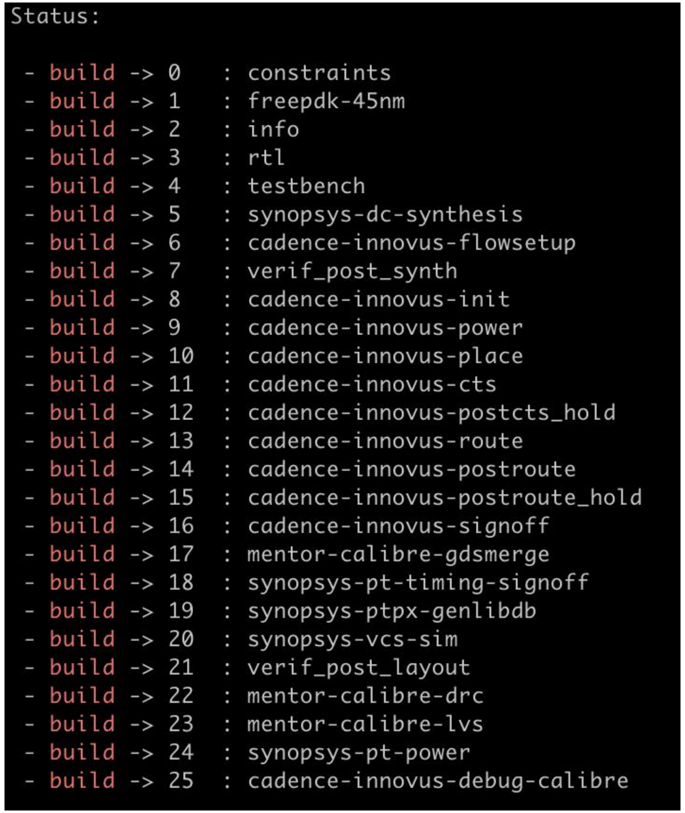
- --> cd /aha/
- --> mflowgen run –demo
- --> cd mflowgen-demo
- --> mkdir build && cd build
- --> mflowgen run --design ../GcdUnit

Caveat: Commercial physical design tools are *not installed*, so this demo will be limited!

mflowgen Demo – Flow runner

Nodes in the graph have numbers (topographical sort)

- Status
  - make status
- Running a node
  - make freepdk-45nm
  - make 1

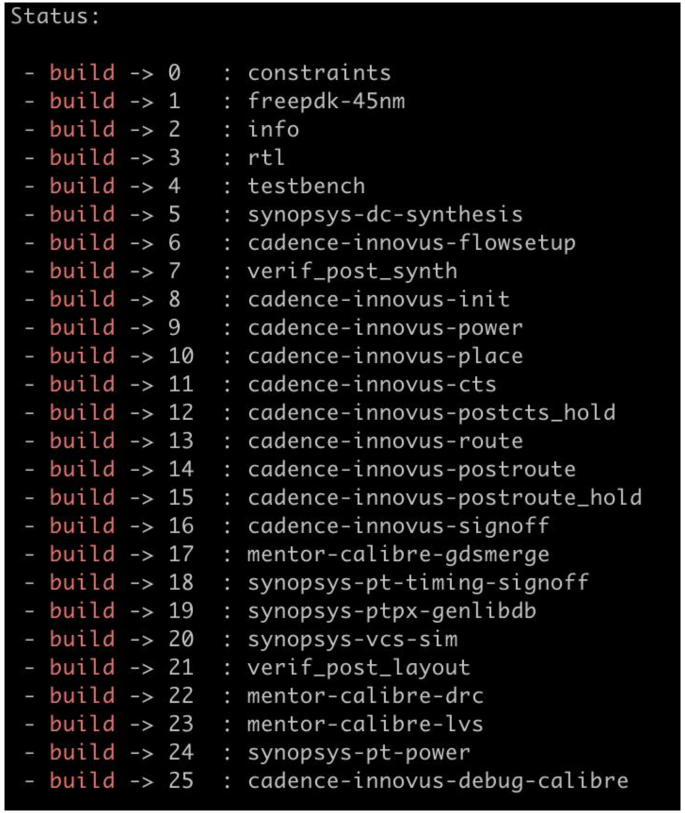


```
Status:

 - build -> 0    : constraints
 - build -> 1    : freepdk-45nm
 - build -> 2    : info
 - build -> 3    : rtl
 - build -> 4    : testbench
 - build -> 5    : synopsys-dc-synthesis
 - build -> 6    : cadence-innovus-flowsetup
 - build -> 7    : verif_post_synth
 - build -> 8    : cadence-innovus-init
 - build -> 9    : cadence-innovus-power
 - build -> 10   : cadence-innovus-place
 - build -> 11   : cadence-innovus-cts
 - build -> 12   : cadence-innovus-postcts_hold
 - build -> 13   : cadence-innovus-route
 - build -> 14   : cadence-innovus-postroute
 - build -> 15   : cadence-innovus-postroute_hold
 - build -> 16   : cadence-innovus-signoff
 - build -> 17   : mentor-calibre-gdsmerge
 - build -> 18   : synopsys-pt-timing-signoff
 - build -> 19   : synopsys-ptpx-genlibdb
 - build -> 20   : synopsys-vcs-sim
 - build -> 21   : verif_post_layout
 - build -> 22   : mentor-calibre-drc
 - build -> 23   : mentor-calibre-lvs
 - build -> 24   : synopsys-pt-power
 - build -> 25   : cadence-innovus-debug-calibre
```

mflowgen Demo – Flow runner

Nodes in the graph have numbers (topographical sort)

- Status
  - make status
- Running a node
  - make freepdk-45nm
  - make 1
- Cleaning a node
  - make clean-1
- Dependencies are reflected
  - make synopsys-dc-synthesis
    - ^ runs freepdk-45nm, rtl, …

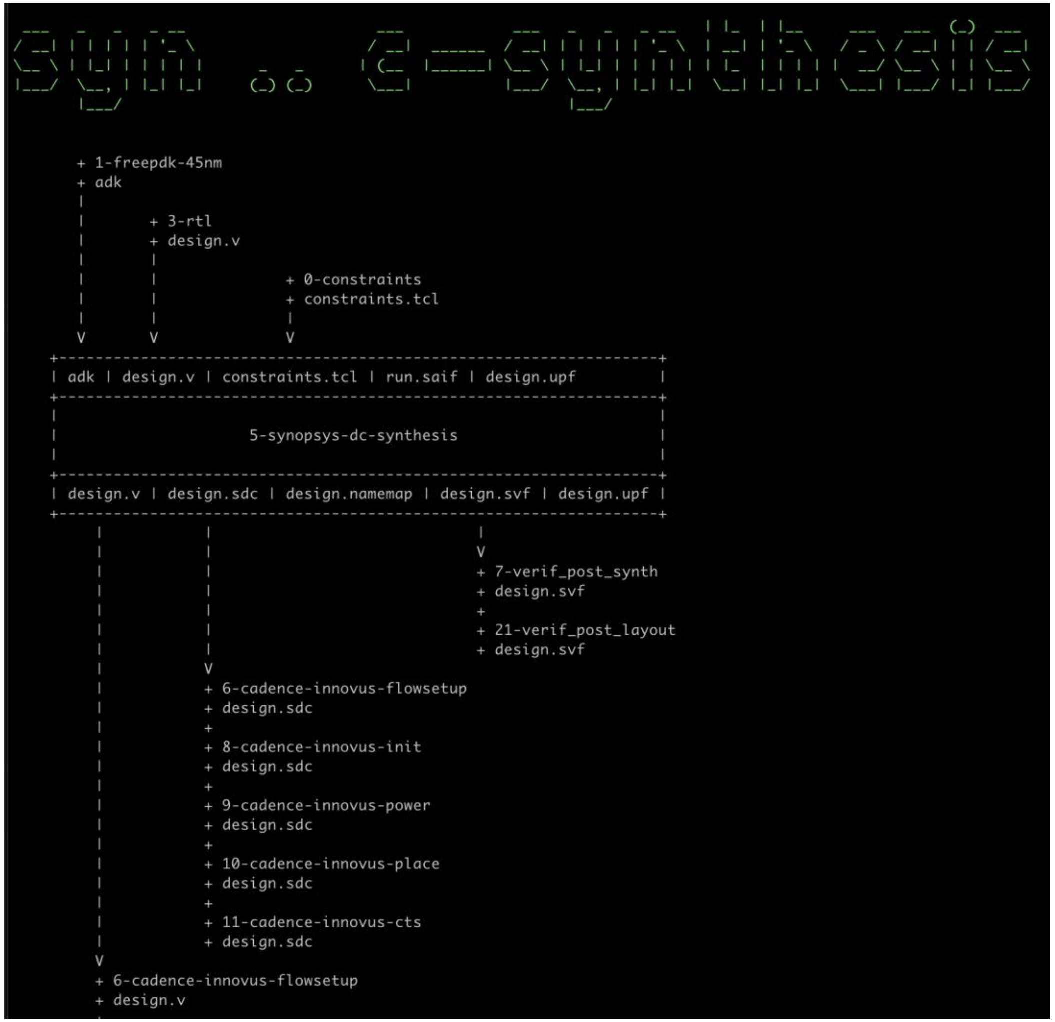


```
Status:

 - build -> 0   : constraints
 - build -> 1   : freepdk-45nm
 - build -> 2   : info
 - build -> 3   : rtl
 - build -> 4   : testbench
 - build -> 5   : synopsys-dc-synthesis
 - build -> 6   : cadence-innovus-flowsetup
 - build -> 7   : verif_post_synth
 - build -> 8   : cadence-innovus-init
 - build -> 9   : cadence-innovus-power
 - build -> 10  : cadence-innovus-place
 - build -> 11  : cadence-innovus-cts
 - build -> 12  : cadence-innovus-postcts_hold
 - build -> 13  : cadence-innovus-route
 - build -> 14  : cadence-innovus-postroute
 - build -> 15  : cadence-innovus-postroute_hold
 - build -> 16  : cadence-innovus-signoff
 - build -> 17  : mentor-calibre-gdsmerge
 - build -> 18  : synopsys-pt-timing-signoff
 - build -> 19  : synopsys-ptpx-genlibdb
 - build -> 20  : synopsys-vcs-sim
 - build -> 21  : verif_post_layout
 - build -> 22  : mentor-calibre-drc
 - build -> 23  : mentor-calibre-lvs
 - build -> 24  : synopsys-pt-power
 - build -> 25  : cadence-innovus-debug-calibre
```

mflowgen Demo

Graph connectivity is easy to access

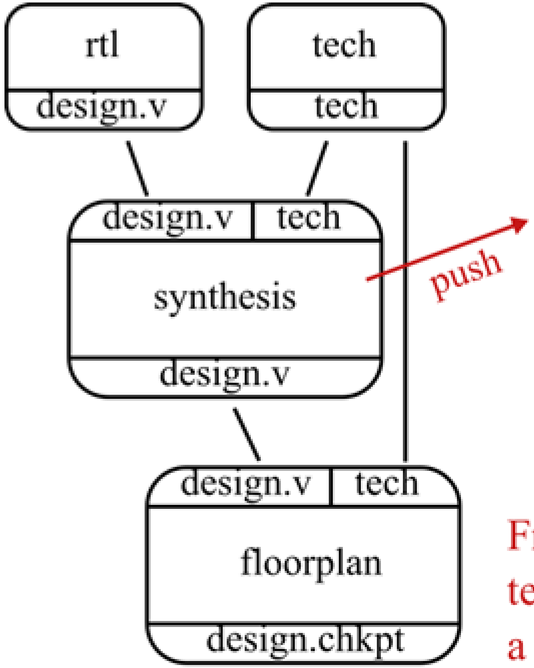
- Zoom in on any node
  - make info-N
  - (E.g., make info-5)
- Corresponds to graph
  - less mflowgen-demo/GcdUnit/construct-commercial.py



```
 ___  _   _  _ __          ___          ___  _   _  _ __  | |_ | |__    ___  ___ (_) ___
/ __|| | | || '_ \        / __| _____  / __|| | | || '_ \ | __|| '_ \  / _ \/ __|| |/ __|
\__ \| |_| || | | | _  _ | (__ |_____| \__ \| |_| || | | || |_ | | | ||  __/\__ \| |\__ \
|___/ \__, ||_| |_|(_)(_) \___|        |___/ \__, ||_| |_| \__||_| |_| \___||___/|_||___/
      |___/                                  |___/

        + 1-freepdk-45nm
        + adk
        |
        |       + 3-rtl
        |       + design.v
        |       |
        |       |              + 0-constraints
        |       |              + constraints.tcl
        |       |              |
        V       V              V
   +-------------------------------------------------------------------+
   | adk | design.v | constraints.tcl | run.saif | design.upf          |
   +-------------------------------------------------------------------+
   |                                                                   |
   |                     5-synopsys-dc-synthesis                       |
   |                                                                   |
   +-------------------------------------------------------------------+
   | design.v | design.sdc | design.namemap | design.svf | design.upf |
   +-------------------------------------------------------------------+
          |           |                             |
          |           |                             V
          |           |                             + 7-verif_post_synth
          |           |                             + design.svf
          |           |                             +
          |           |                             + 21-verif_post_layout
          |           |                             + design.svf
          |           V
          |           + 6-cadence-innovus-flowsetup
          |           + design.sdc
          |           +
          |           + 8-cadence-innovus-init
          |           + design.sdc
          |           +
          |           + 9-cadence-innovus-power
          |           + design.sdc
          |           +
          |           + 10-cadence-innovus-place
          |           + design.sdc
          |           +
          |           + 11-cadence-innovus-cts
          |           + design.sdc
          V
          + 6-cadence-innovus-flowsetup
          + design.v
```

mflowgen Demo - Sharing within a team

Nodes are natural candidates for **sharing pre-built checkpoints**



Stashed Nodes

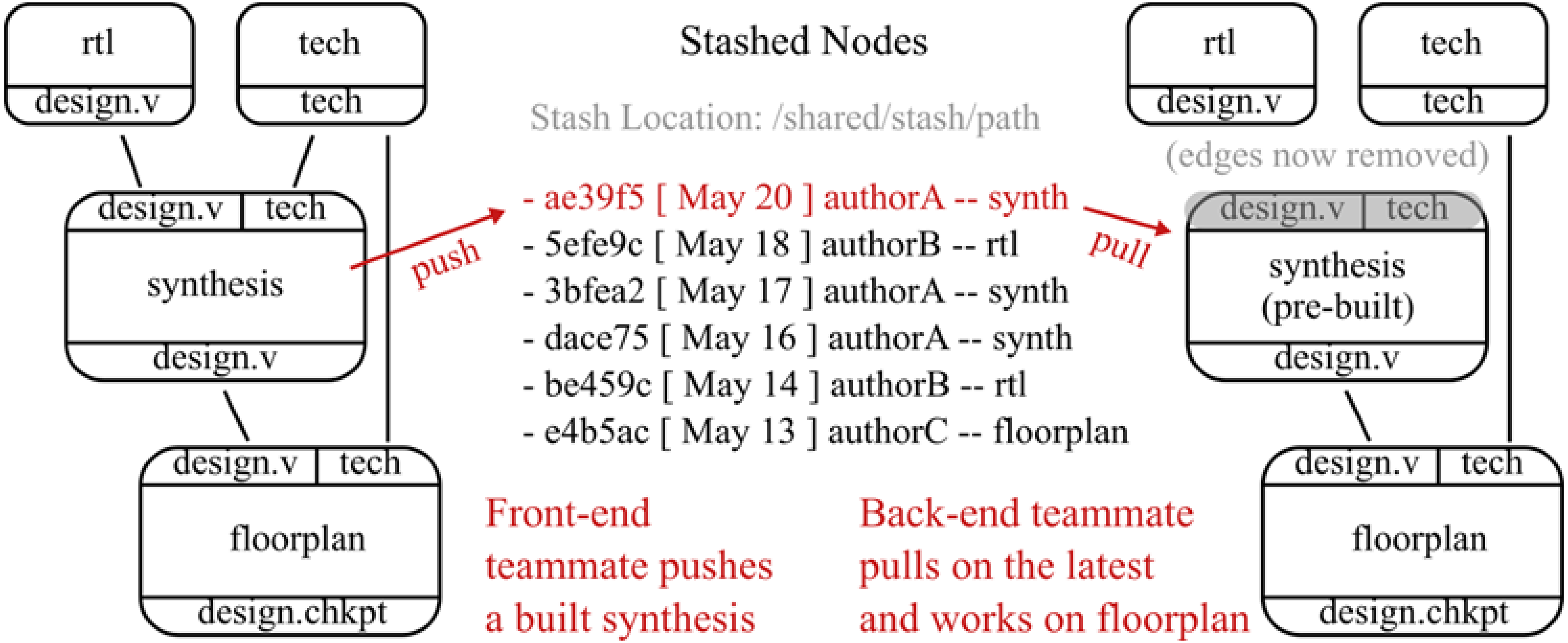
Stash Location: /shared/stash/path

- ae39f5 [May 20] authorA -- synth
- 5efe9c [May 18] authorB -- rtl
- 3bfea2 [May 17] authorA -- synth
- dace75 [May 16] authorA -- synth
- be459c [May 14] authorB -- rtl
- e4b5ac [May 13] authorC -- floorplan

Front-end
teammate pushes
a built synthesis

mflowgen Demo - Sharing within a team

Nodes are natural candidates for **sharing pre-built checkpoints**



Stashed Nodes

Stash Location: /shared/stash/path

```
- ae39f5 [ May 20 ] authorA -- synth
- 5efe9c [ May 18 ] authorB -- rtl
- 3bfea2 [ May 17 ] authorA -- synth
- dace75 [ May 16 ] authorA -- synth
- be459c [ May 14 ] authorB -- rtl
- e4b5ac [ May 13 ] authorC -- floorplan
```

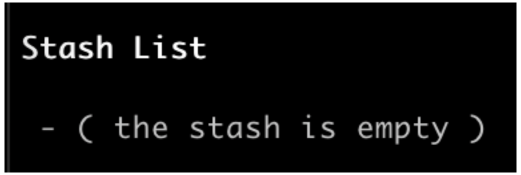
Front-end teammate pushes a built synthesis

Back-end teammate pulls on the latest and works on floorplan

mflowgen Demo - Sharing within a team

Nodes are natural candidates for **sharing pre-built checkpoints**

- Initialize an mflowgen stash
    - cd build
    - mflowgen stash init -p ../
    - mflowgen stash list
        - ^ empty stash

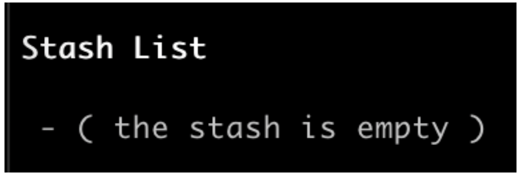


```
Stash List

 - ( the stash is empty )
```

mflowgen Demo - Sharing within a team

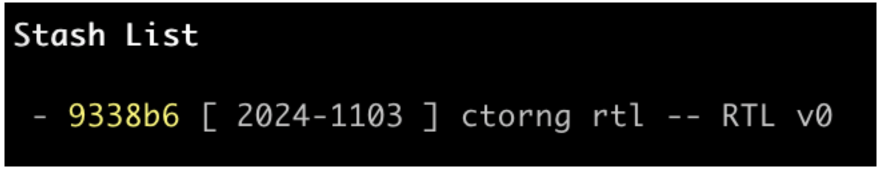
Nodes are natural candidates for **sharing pre-built checkpoints**

- Initialize an mflowgen stash
  - cd build
  - mflowgen stash init -p ../
  - mflowgen stash list
    - ^ empty stash



```
Stash List

 - ( the stash is empty )
```

- Push built node to the stash
  - make 3 # 3 is the rtl node
  - mflowgen stash push --step 3 -m "RTL v0”
  - mflowgen stash list



```
Stash List

 - 9338b6 [ 2024-1103 ] ctorng rtl -- RTL v0
```

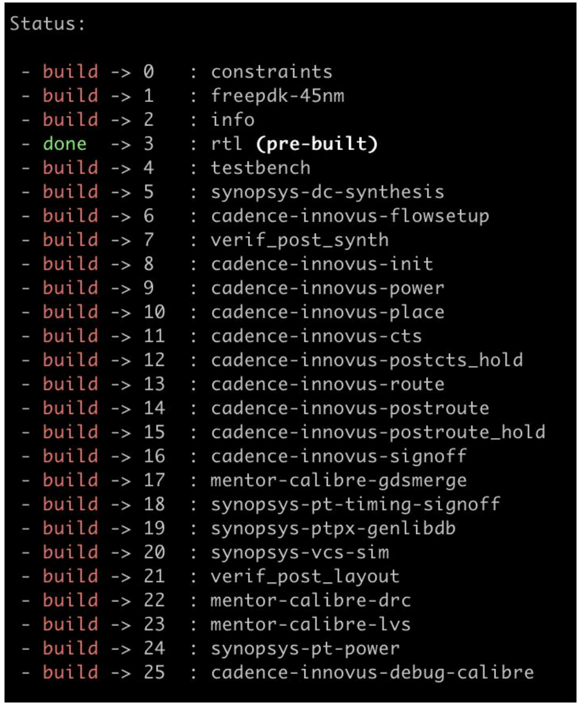
Hash **Date** **Author** **Node and Message**

mflowgen Demo - Sharing within a team

Nodes are natural candidates for **sharing pre-built checkpoints**

Acting as someone else …

- Link other build to an existing mflowgen stash
  - cd ..
  - mkdir build-2 && cd build-2
  - mflowgen stash link -p ../2024-1103-mflowgen-stash-179441/



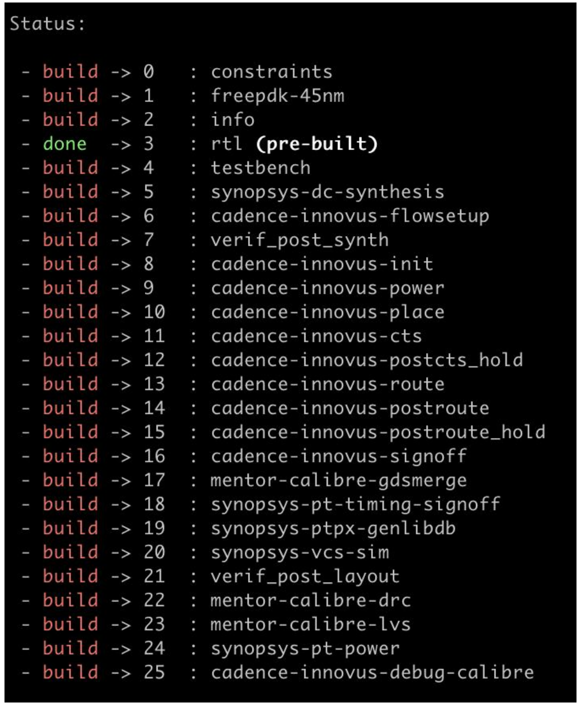
```
Status:

 - build -> 0   : constraints
 - build -> 1   : freepdk-45nm
 - build -> 2   : info
 - done  -> 3   : rtl (pre-built)
 - build -> 4   : testbench
 - build -> 5   : synopsys-dc-synthesis
 - build -> 6   : cadence-innovus-flowsetup
 - build -> 7   : verif_post_synth
 - build -> 8   : cadence-innovus-init
 - build -> 9   : cadence-innovus-power
 - build -> 10  : cadence-innovus-place
 - build -> 11  : cadence-innovus-cts
 - build -> 12  : cadence-innovus-postcts_hold
 - build -> 13  : cadence-innovus-route
 - build -> 14  : cadence-innovus-postroute
 - build -> 15  : cadence-innovus-postroute_hold
 - build -> 16  : cadence-innovus-signoff
 - build -> 17  : mentor-calibre-gdsmerge
 - build -> 18  : synopsys-pt-timing-signoff
 - build -> 19  : synopsys-ptpx-genlibdb
 - build -> 20  : synopsys-vcs-sim
 - build -> 21  : verif_post_layout
 - build -> 22  : mentor-calibre-drc
 - build -> 23  : mentor-calibre-lvs
 - build -> 24  : synopsys-pt-power
 - build -> 25  : cadence-innovus-debug-calibre
```

mflowgen Demo - Sharing within a team

Nodes are natural candidates for **sharing pre-built checkpoints**

Acting as someone else …

- Link other build to an existing mflowgen stash
  - cd ..
  - mkdir build-2 && cd build-2
  - mflowgen stash link -p ../2024-1103-mflowgen-stash-179441/
- Pull built node from an mflowgen stash
  - cd build
  - mflowgen stash pull --hash 9338b6

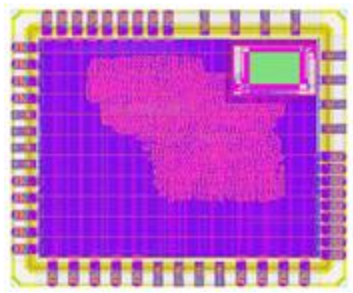


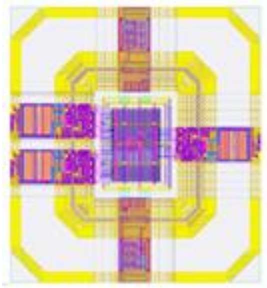
```
Status:

 - build -> 0   : constraints
 - build -> 1   : freepdk-45nm
 - build -> 2   : info
 - done  -> 3   : rtl (pre-built)
 - build -> 4   : testbench
 - build -> 5   : synopsys-dc-synthesis
 - build -> 6   : cadence-innovus-flowsetup
 - build -> 7   : verif_post_synth
 - build -> 8   : cadence-innovus-init
 - build -> 9   : cadence-innovus-power
 - build -> 10  : cadence-innovus-place
 - build -> 11  : cadence-innovus-cts
 - build -> 12  : cadence-innovus-postcts_hold
 - build -> 13  : cadence-innovus-route
 - build -> 14  : cadence-innovus-postroute
 - build -> 15  : cadence-innovus-postroute_hold
 - build -> 16  : cadence-innovus-signoff
 - build -> 17  : mentor-calibre-gdsmerge
 - build -> 18  : synopsys-pt-timing-signoff
 - build -> 19  : synopsys-ptpx-genlibdb
 - build -> 20  : synopsys-vcs-sim
 - build -> 21  : verif_post_layout
 - build -> 22  : mentor-calibre-drc
 - build -> 23  : mentor-calibre-lvs
 - build -> 24  : synopsys-pt-power
 - build -> 25  : cadence-innovus-debug-calibre
```

Stanford University

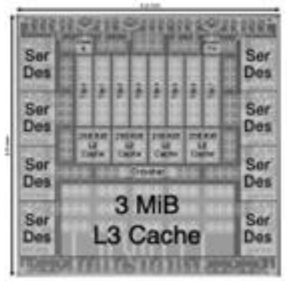
Takeaway: Accessibility of PD is expanding through agile flow tools

SiliconCompiler

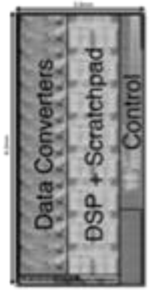

SKY130


SKY130

Hammer (dozens fabricated chips in class)



16nm

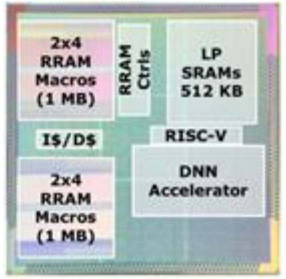


22nm

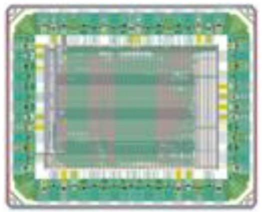
12nm

mflowgen (dozens fabricated chips in class)

16nm



40nm


28nm

- **Students touched**: Stanford, Berkeley, Cornell, Georgia Tech, USC, etc…
- **Industry**: SiliconCompiler investment "build to scale" and "plan for 10+ years"

[*] https://github.com/the-openroad-project

mflowgen - Integration with OpenROAD

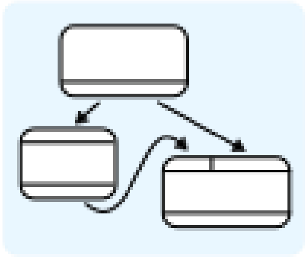
Now integrated with OpenROAD:

- orfs-yosys-synthesis
- orfs-openroad-floorplan
- orfs-openroad-place
- orfs-openroad-cts
- orfs-openroad-route
- orfs-openroad-finish
- orfs-docker-setup

Lays the groundwork for **highly accessible open-source chip design flows** for student learners

Documentation for mflowgen + OpenROAD: https://mflowgen.readthedocs.io/en/latest/stdlib-openroad.html

mflowgen Takeaways



mflowgen developed since 2018 at Stanford + USC

- **Development cost**: Exceeds $1M, 50+ person effort
- ~1200 commits, 20K+ lines Python/Tcl
- 25+ classroom tapeouts (non-research) by 100+ students (2020-2024)

- **Six chip tapeout courses** at Stanford (2020-2023) + USC (2023-2024)

- **Three chip tapein courses** at Cornell (2021-2023)

- **Technologies**: GF12, Intel16, TSMC16, TSMC28, TSMC40, IBM130. SKY130, and IBM180

- **Github and ReadtheDocs**: 1000+ commits currently, active docs