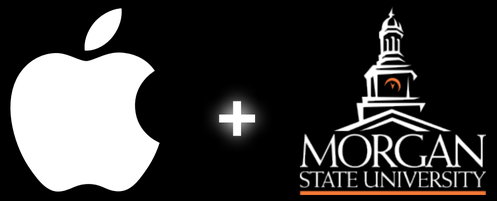


Bring-Up Best Practices

Guest lecture

Eric Smith
11/14/23

Recruiting Contact: nsi@apple.com

Who is Eric

- Electrical engineer who started at Apple in 2005
  - Previous employer was Teradyne (2000-2005)
  - Specialized in parametric measurement
- Lead the Mobile Mac MLB team until 2010
- Architected power supply ASICs until 2015
- Recent focus: internal sensors, control system design, VLSI CAD tooling

Goals of Talk

- Introduce some industry terms in test.
- Introduce common commercial test techniques.
- Provide some helpful hints you can use in your design.

“If it hasn’t been tested, it doesn’t work.”

Digital Design a Systems Approach
Dally & Harting
Page 27

Common industry terms

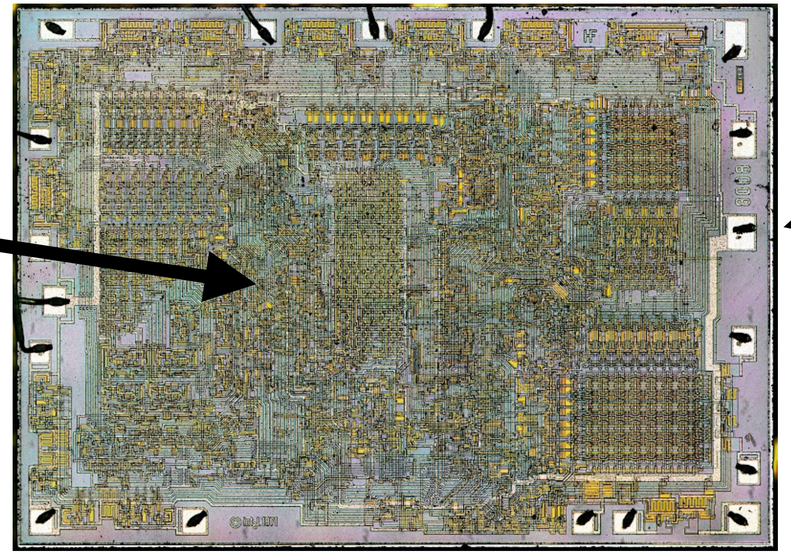
Term	Definition
Validation	Have you designed the **right thing**? (Is the spec correct)
Verification	Have you designed the **thing right**? (Does it work?)
Pre-Silicon	Tests and checks done in a virtual environment, or against databases, before the IC is constructed. (Easy to fix)
Post-Silicon	Tests and checks done in a lab on the part. (Expensive to fix)
Qualification	Tests to ensure survival of the design in the environment it is to be used in.

Term	Definition
ATE	Automated Test Equipment
DUT	Device Under Test
Wafer Test	Using ATE on a DUT still located on a wafer.
System Test	Tests on a DUT after it has been placed inside a package.
Parametric Test	A line item in a datasheet is a parameter (rise time, IDDQ, etc.). These are the checks on those.
ATPG	Automated Test Pattern Generation (Generator)
BIST	Built-in Self Test

Why have this talk at the beginning of the course?

When testing ASICs we have a fundamental problem:

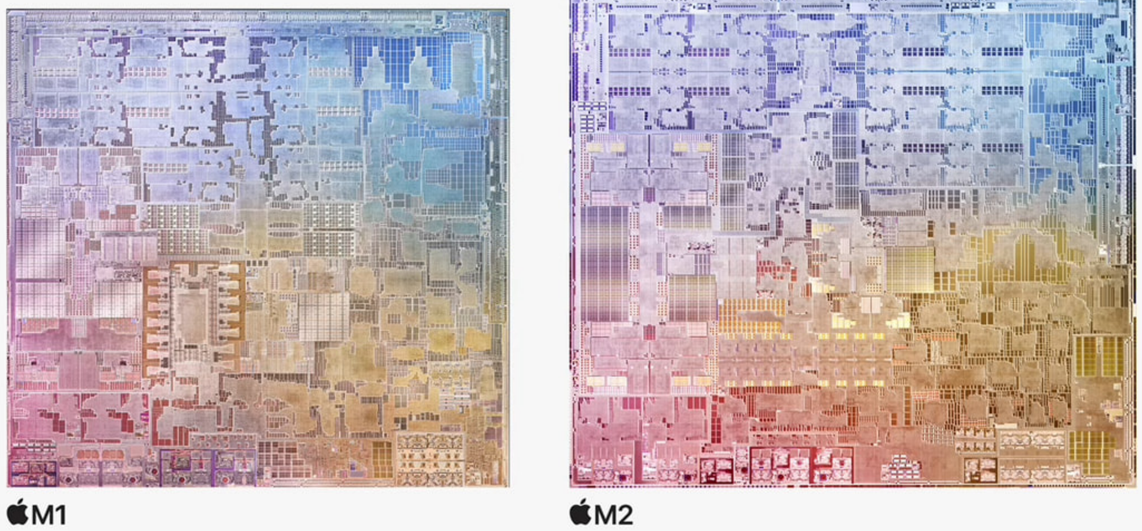
Lots of stuff that we would like to measure



Not so many pins connect the outside world to the inside world.

To get around this, debug features need to be planned for and designed in.

In 2023, ASICs have a lot more stuff to test



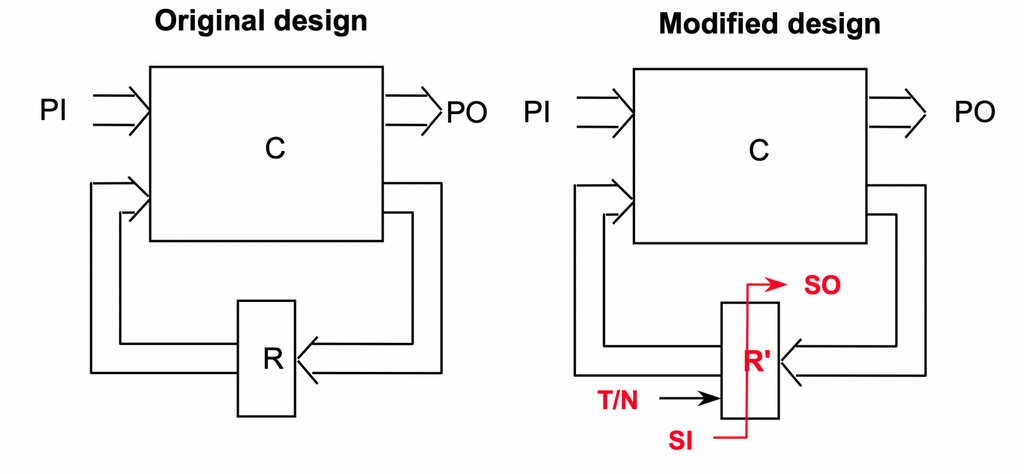
https://www.apple.com/newsroom/2022/06/apple-unveils-m2-with-breakthrough-performance-and-capabilities/

This requires planning and infrastructure.

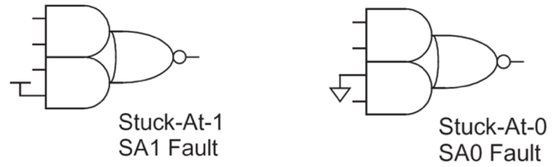
Common Approaches for Dealing with the Dilemma

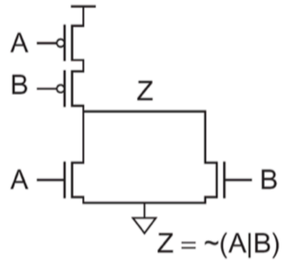
- Scan-Based Testing
- Built-in Self Test (BIST)
- Ad Hoc Testing

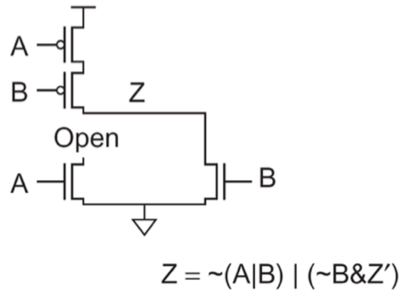
Scan-Based Testing Concepts (ATPG)



https://eecs.ceas.uc.edu/~jonewb/intro.pdf







http://pages.hmc.edu/harris/cmosvlsi/4e/figures/cmos15.ppt

Scan-Based Testing

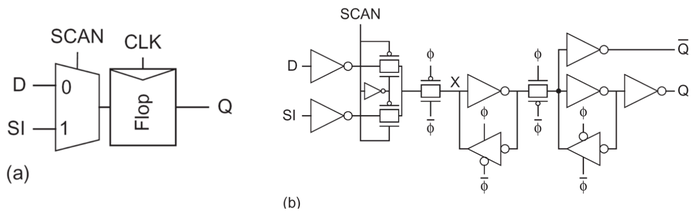
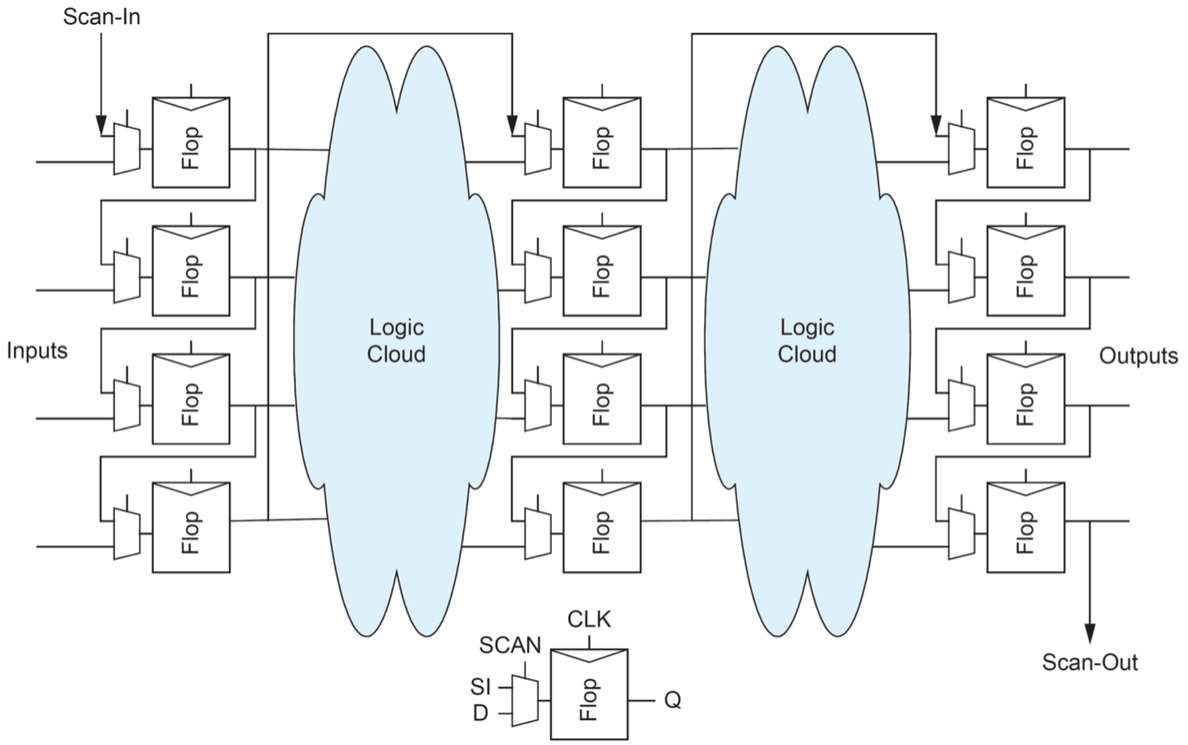


FIGURE 15.18 Scannable flip-flops

http://pages.hmc.edu/harris/cmosvlsi/4e/figures/cmos15.ppt

FIGURE 15.16 Scan-based testing

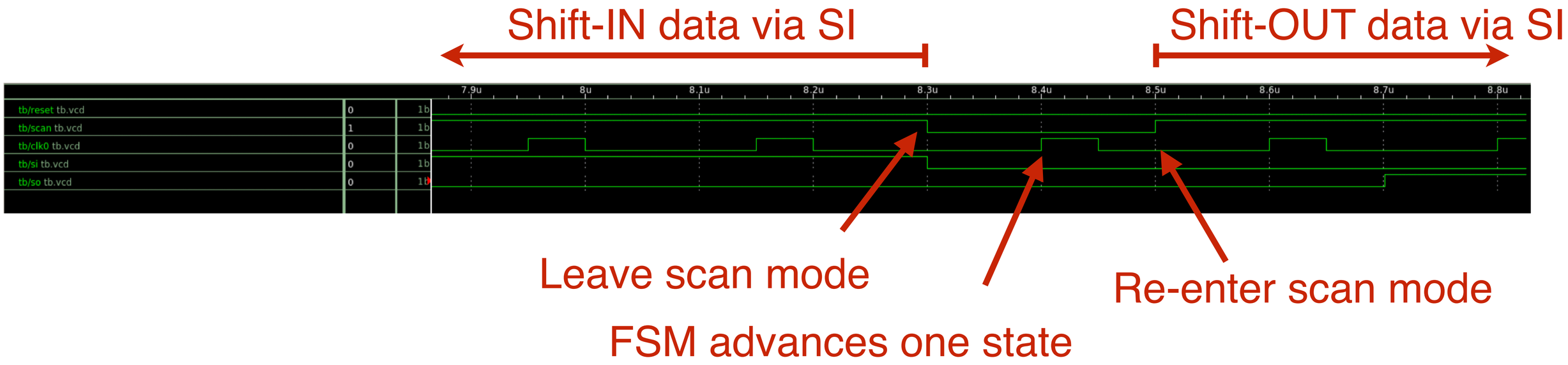


Automatic Test Pattern Generation (ATPG)

https://en.wikipedia.org/wiki/Automatic_test_pattern_generation

Scan-Based Testing

Partial Scan Sequence



Scan-Based Testing Concepts (ATPG)

If you want more information on how the ATPG software works check out these resources.

EECS224 Fall 2005

http://people.eecs.berkeley.edu/~keutzer/classes/244fa2005/244fa2005-h6.htm

Prof. Kurt Keutzer

Logic Synthesis
S. Devadas, A. Ghosh, K. Keutzer, McGraw Hill
1994

Chapters 5 and 9

https://iccad.com

Q: But what controls the scan chain?
A: The TAP (Test Access Protocol) Architecture

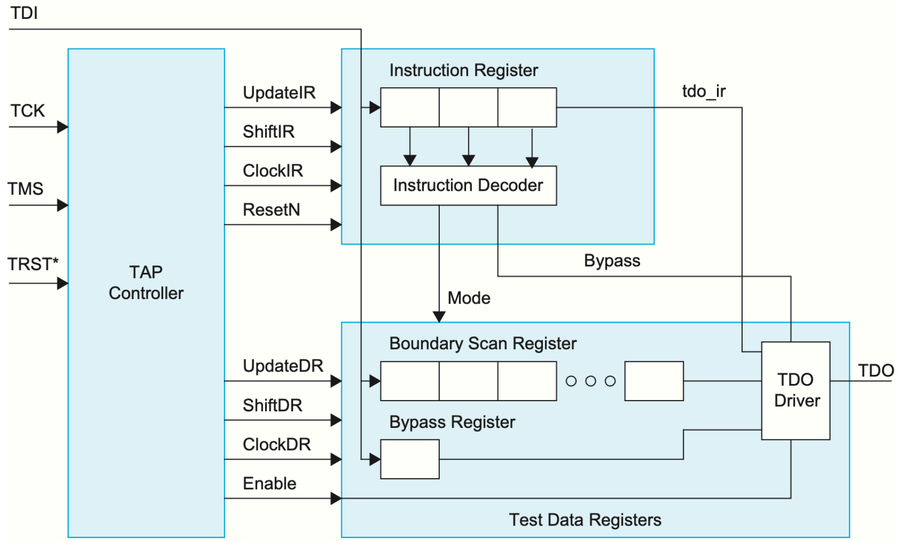


FIGURE W15.6 TAP architecture

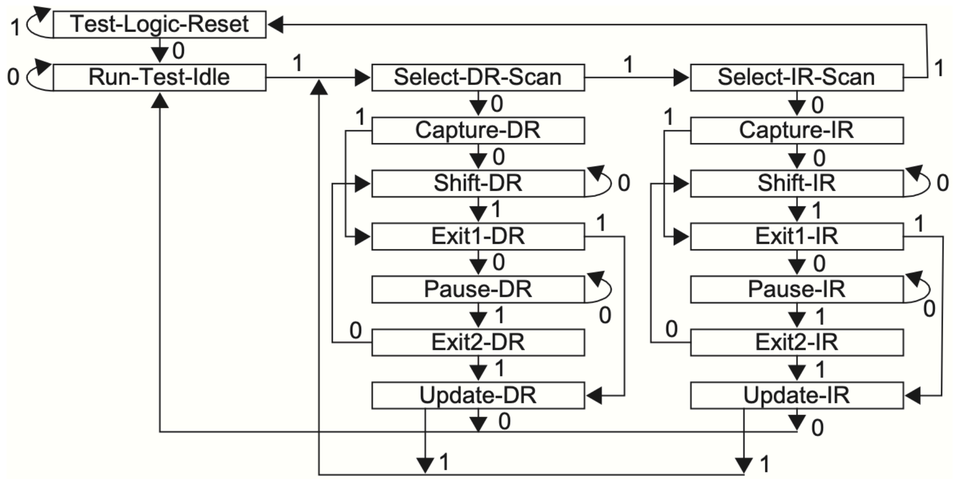


FIGURE W15.7 TAP controller state diagram

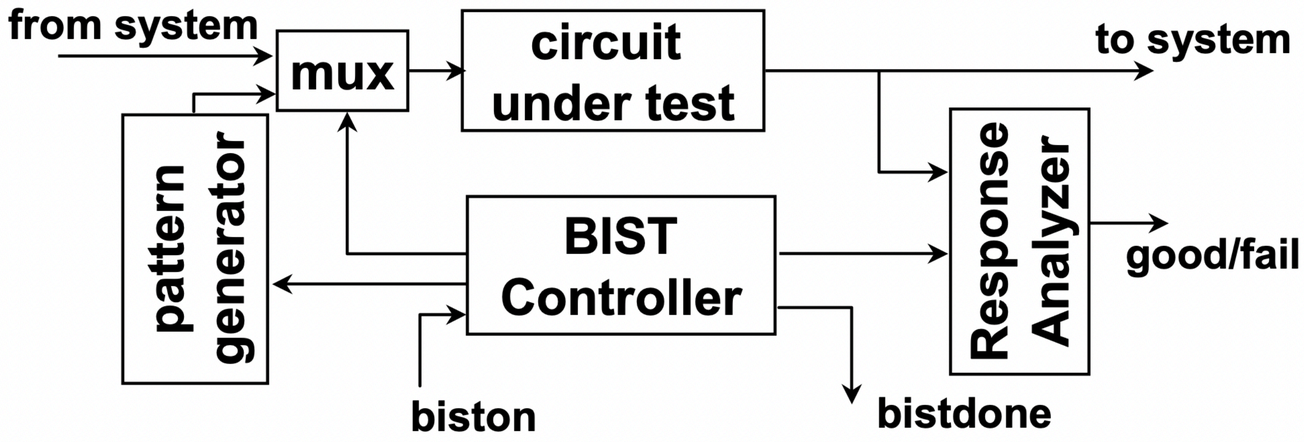
Too much to cover for this lecture. Please follow link below for more information.

http://pages.hmc.edu/harris/cmosvlsi/4e/Westeweb.fm.pdf

Built-in Self Test

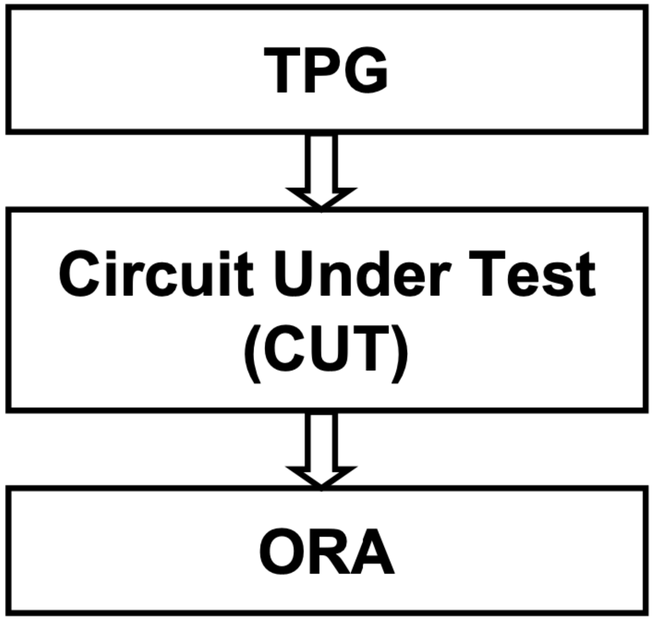
Built-In-Self Test (BIST)

- Places the job of device testing inside the device itself
- Generates its own stimulus and analyzes its own response



https://eecs.ceas.uc.edu/~jonewb/intro.pdf

The Basic Architecture of Built-in Self Test (BIST)



https://eecs.ceas.uc.edu/~jonewb/intro.pdf

TPG: Test pattern generator
ORA Output response analyzer

Built-in Self Test (BIST)

Actually, go here.

https://eecs.ceas.uc.edu/~jonewb/

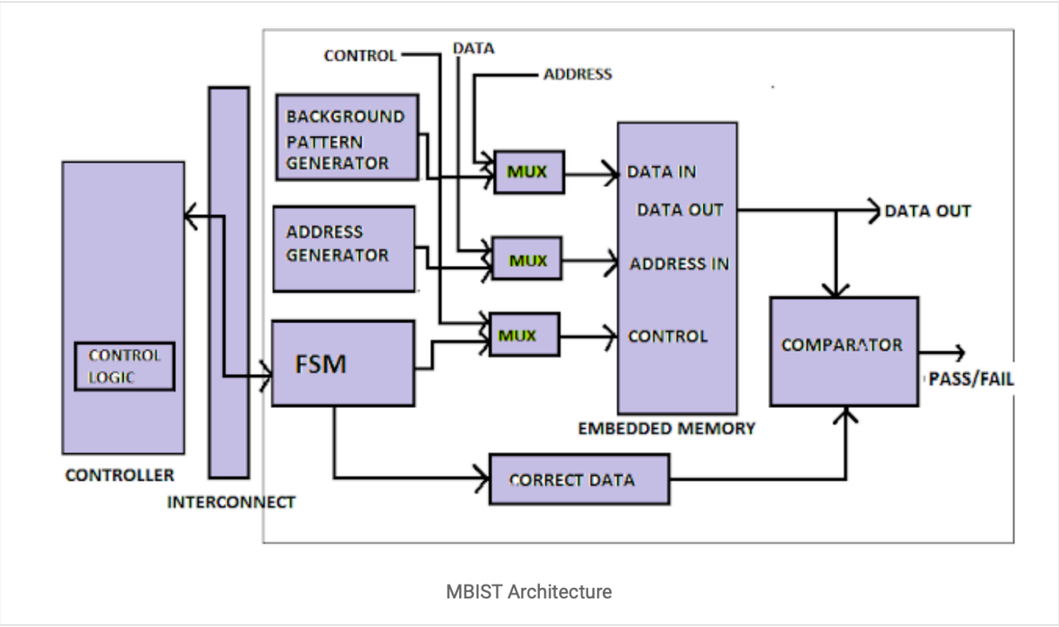
Dr. Wen-Ben Jone has many excellent slides from his VLSI testing course.

To Learn More About Logic BIST (LBIST):

https://cdnc.itec.kit.edu/downloads/tds2_ws1011_lecture7.pdf

https://eecs.ceas.uc.edu/~jonewb/BIST2.pdf

Another very important area is Memory BIST (MBIST)



MBIST Architecture

https://www.vlsi4freshers.com/2019/12/memory-built-in-self-test-mbist-basic.html

Ad Hoc Testing

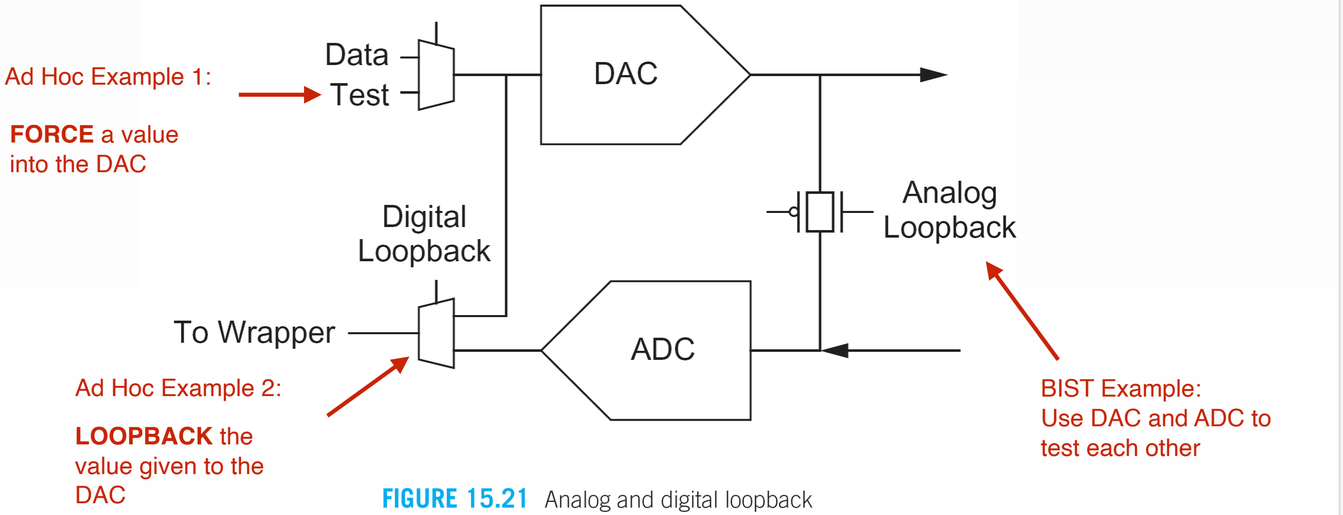


FIGURE 15.21 Analog and digital loopback

http://pages.hmc.edu/harris/cmosvlsi/4e/figures/cmos15.ppt

Ad Hoc means “for this.” It’s the catch-all category and represents the tricks and institutional knowledge teams have. It’s often done with muxes to force and read critical values in a system.

What makes bring-up different than design validation?

Bring up is about confirming the most fundamental assumptions you made when you created the design.

1. Are pins connected?
2. Is a module powered?
3. Is clock being distributed?
4. Can the system enter reset?
5. Etc.

It’s also about confirming the most fundamental modules in a design. These are the ones that have to function before verification of the total design can begin.

An example of a fundamental modules

The ARM Processor is very important.

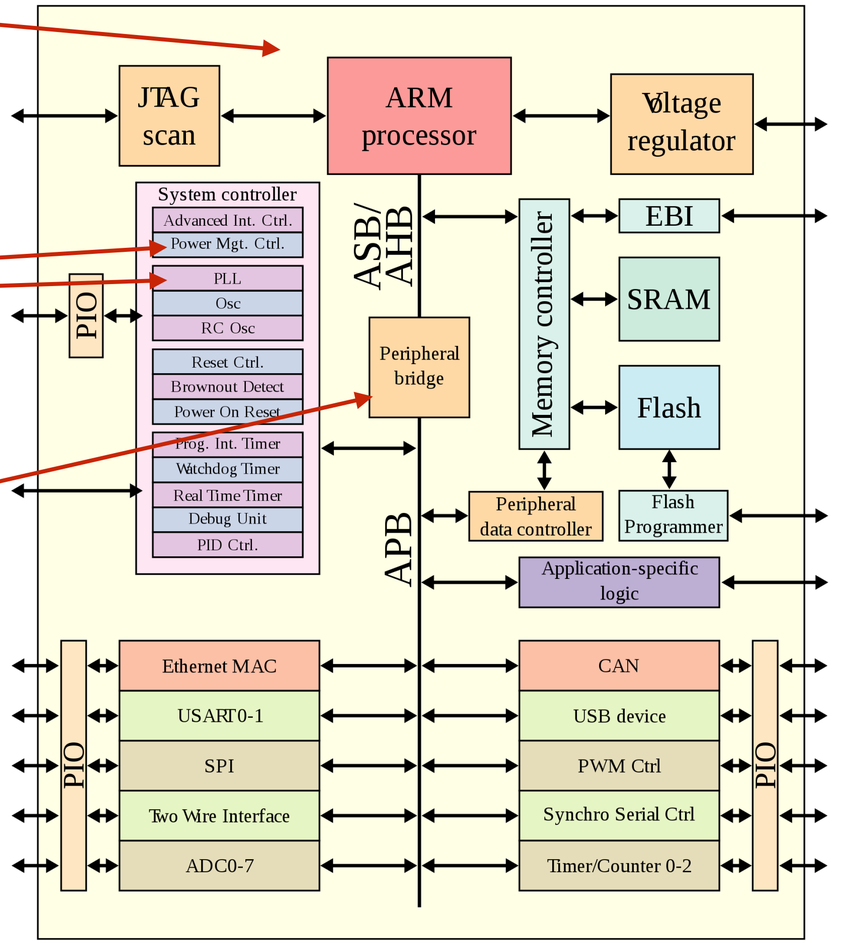
If it fails the chip will not be successful. But it's not fundamental because the SOC can do things without it.

Power management and clocks are fundamental

Therefore they can be directly controlled by dedicated PIO.

The Peripheral Bridge is fundamental

No module can receive any data, including test data, without it.

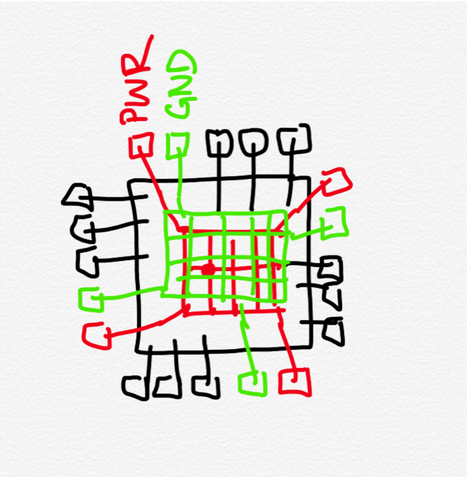


https://en.wikipedia.org/wiki/System_on_a_chip#/media/File:ARMSoCBlockDiagram.svg

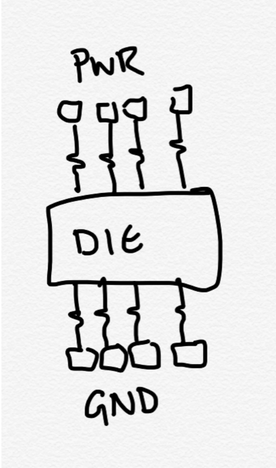
When making a bring-up plan: Identify the fundamental things

1. What needs to work for your team to validate the rest of the design.
    1. Power
    2. Clocks
    3. System Bus
    4. etc.
2. Try to make the number of fundamental things as small as you can.
3. Make sure you simulate the bring-up process!
4. If a fundamental thing doesn’t work, how will you interrogate the issue?
   Have a plan in place for observability.

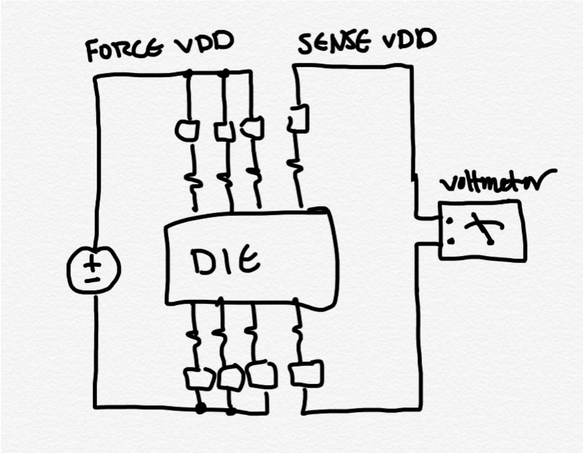
Bring-Up Best Practice #1 : Power Sense Pins



How to measure an internal grid voltage?



If we think of the equivalent circuit, the power pins themselves tap the grid.

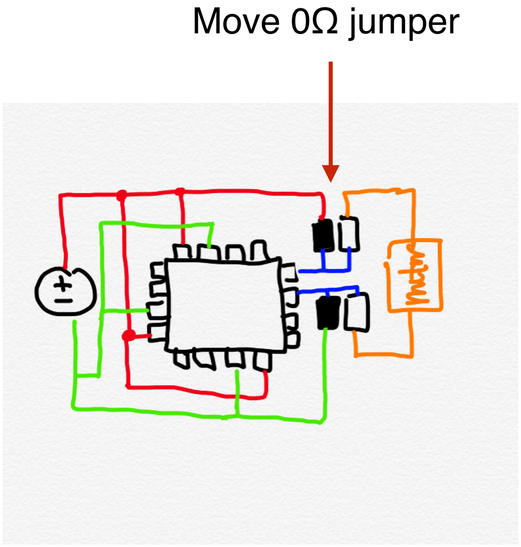


Use some pins to supply power and some to sense voltage.

This is a Kelvin Sense and the extra pin is being used as a Kelvin Contact.

A very important lab technique.

https://en.wikipedia.org/wiki/Four-terminal_sensing



Loosing a power distribution pin is not great. So after bring-up, let it contribute!

Many companies have dedicated bring-up PCBs with layouts slightly different than production to aid bring-up.

Bring-Up Best Practice #2 : How to validate the clock?

Problems:

1. The clock is very fast
2. The clock tree has many leaves

Solutions involve trade-offs:

1. Loose some visibility
2. Add more debug circuits

    This costs: complexity and area.

Timing Uncertainty Measurements on the Power5 Microprocessor
Phillip J. Restle, Robert L. Franch
ISSCC 2004

An example of clock distribution

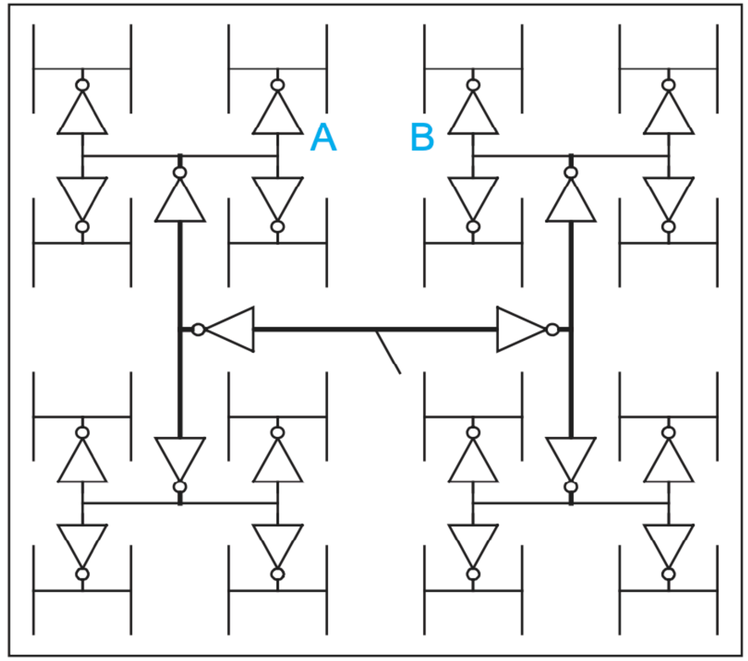
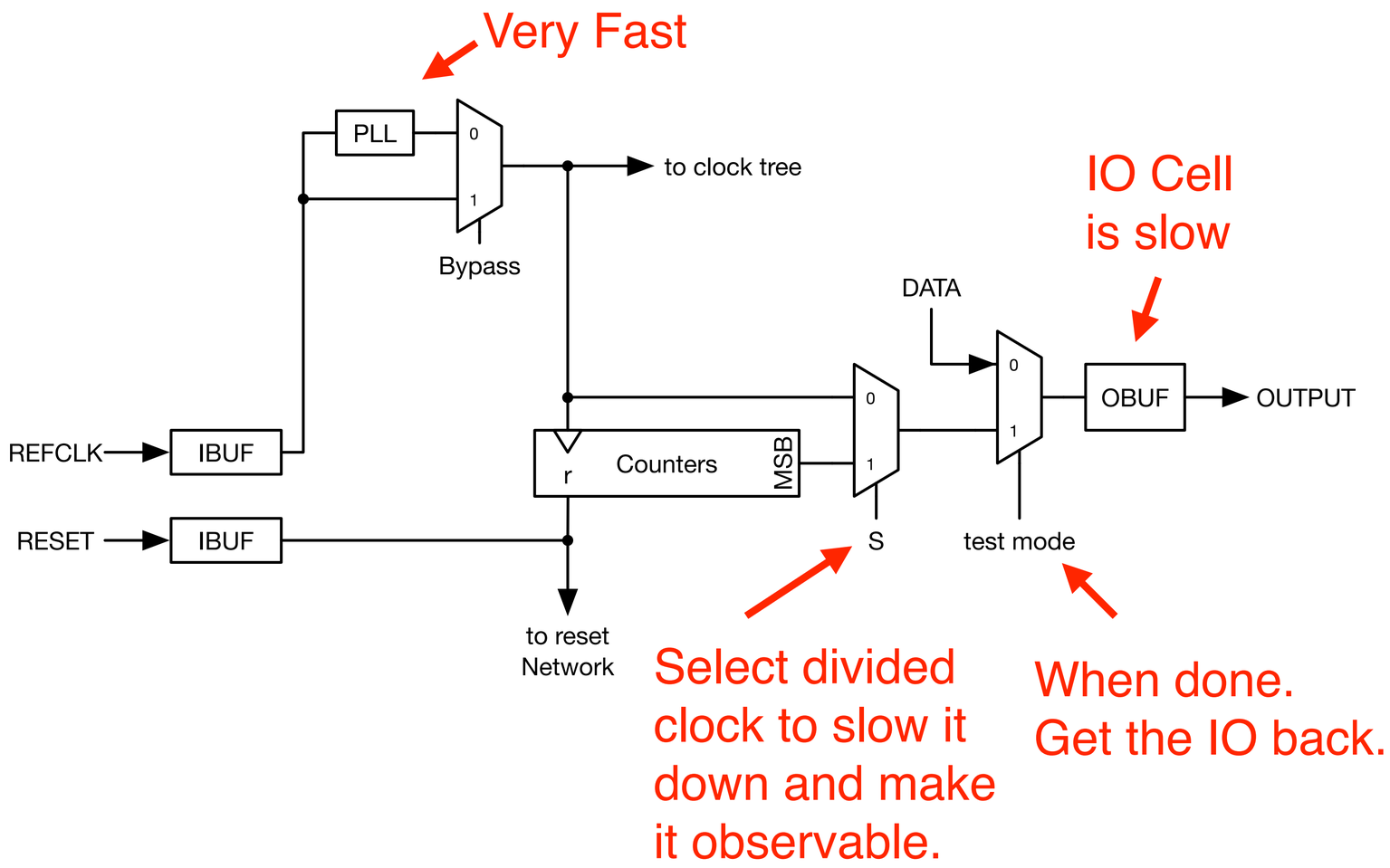


FIGURE 13.24 H-tree

http://pages.hmc.edu/harris/cmosvlsi/4e/figures/cmos13.ppt

Bring-Up Best Practice #2 : Send out a divided clock



Beware trap for new-players (EEVBlog)

If clocks are broken the state of bypass, S and testmode probably can't be changed.

Make sure power-on defaults work toward your plan.

i.e. bypass = 1, S=0, testmode=1

Now REFCLK has a combinational path out of the ASIC.

Bring-Up Best Practice #3 : Testing SystemBus Itself

Problem:

In general the SystemBus has:

1. Very low latency
2. Very high bandwidth
3. Tons of wires everywhere

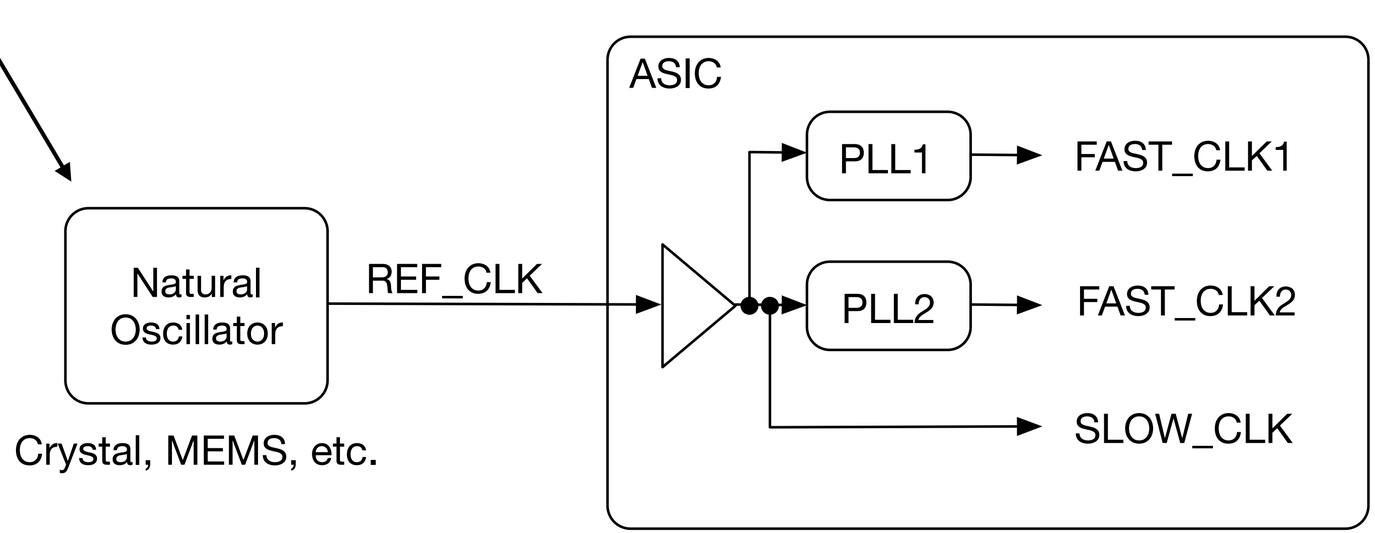
Observing it on the outside of the ASIC is not easy.

This leads to the verification engineers best trick: **Time Dilation**

Bring-Up Best Practice #3 : Clock DUT slowly

The SystemBus is too fast, but relative to what?
The chip doesn’t actually know what time is.

The ASIC defines a second based on the physical construction of this device.



Bring-Up Best Practice #3 : Clock DUT slowly

When debugging, source REF_CLK from something we control.

It doesn't matter how fast REF_CLK is.
What matters is the ratio:

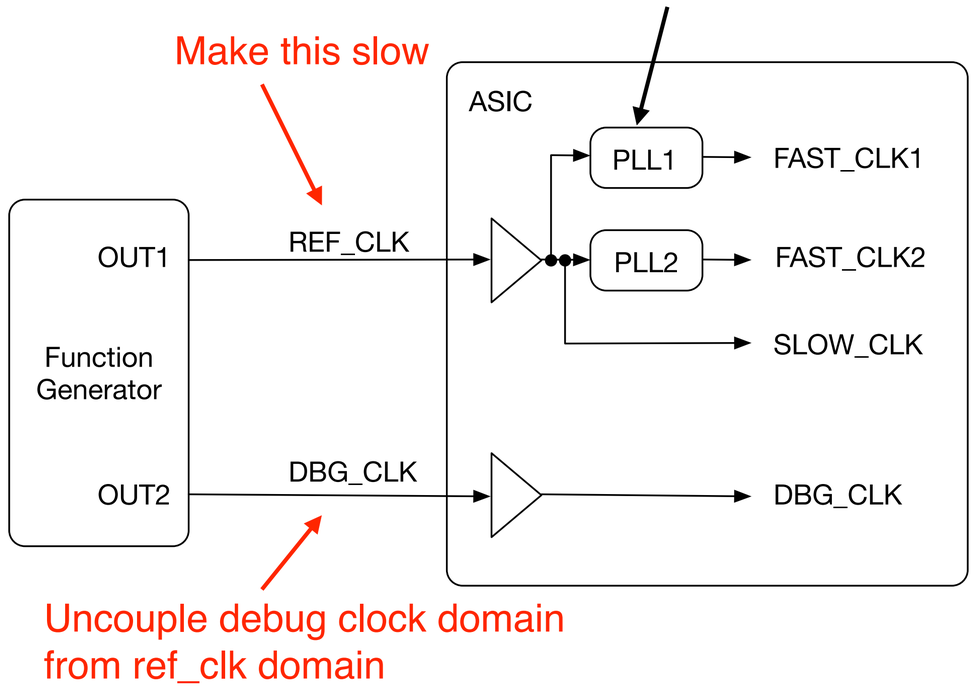
$$\frac{DBG\_CLK}{REF\_CLK}$$

For example:
DBG_CLK : REF_CLK :: 10MHz : 10KHz

This means the debug system can take 1000 actions for every 1 action of the rest of the system.

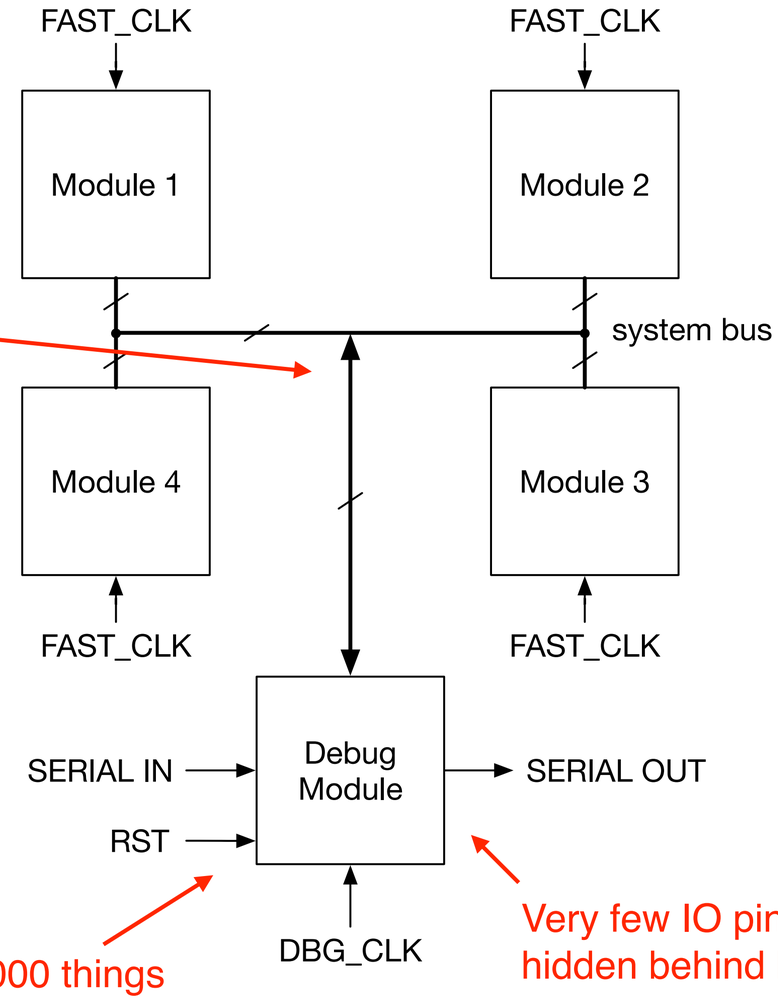
Possible TRAP:

If REF_CLK is too slow the PLL might not lock as it requires feedback at a minimum interval. So make sure you can bypass them.



Bring-Up Best Practice #3 : Clock DUT slowly

Assume
DBG_CLK : REF_CLK :: 10MHz : 10KHz



The debug module can observe and inject packets into the system bus.

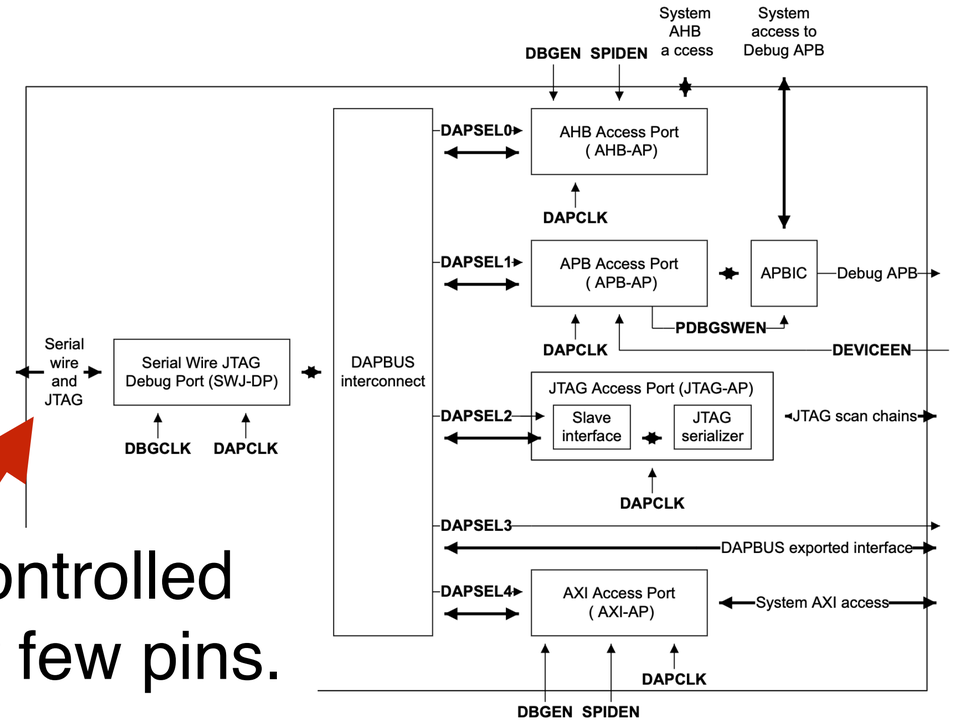
Now this module can to do ~1000 things every FAST_CLK cycle.

Very few IO pins, and they can be hidden behind MUXs to reclaim IO.

Bring-Up Best Practice #3 : Clock DUT slowly

An example how the ARM system does it.

Debug Module

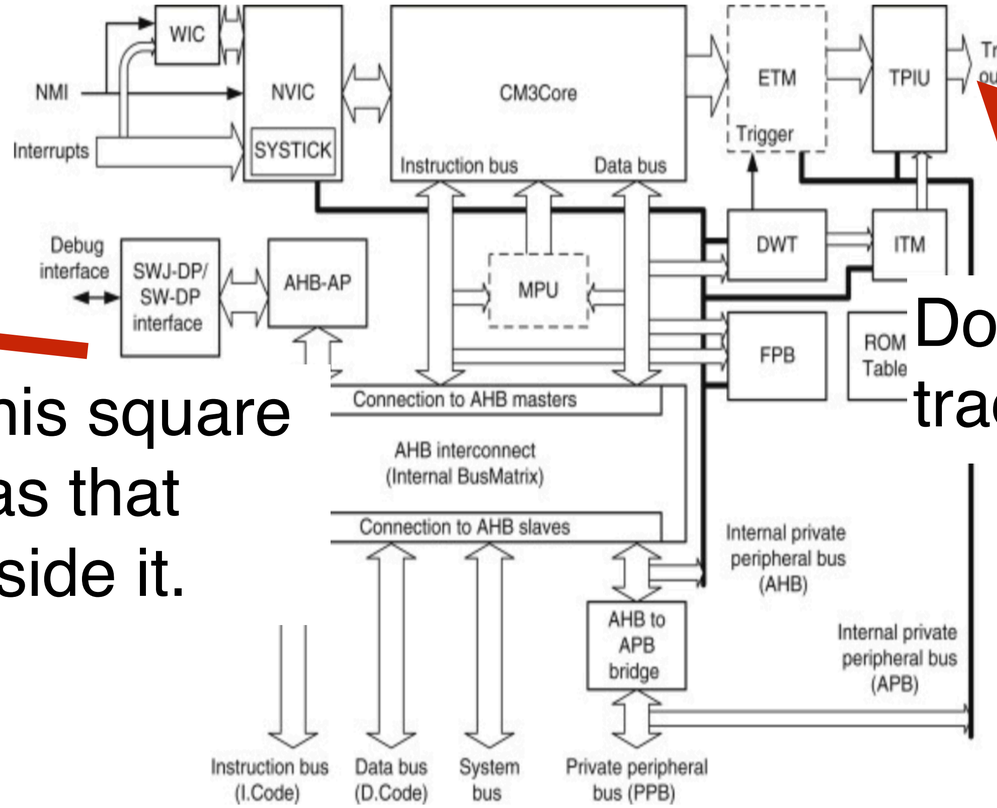


Controlled by few pins.

https://documentation-service.arm.com/static/5f87365c405d955c5176e73e?token=

Make this relatively fast

Typical Core M3 Debug Architecture



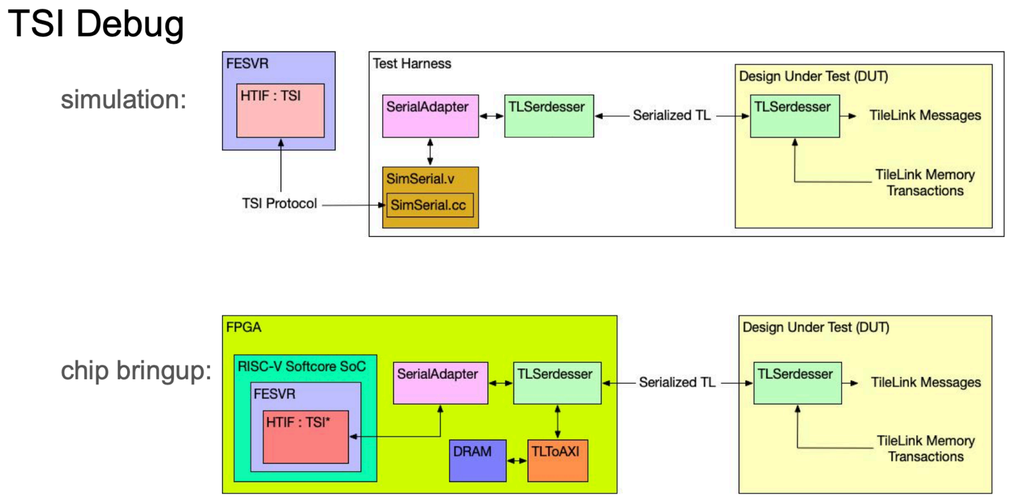
This square has that inside it.

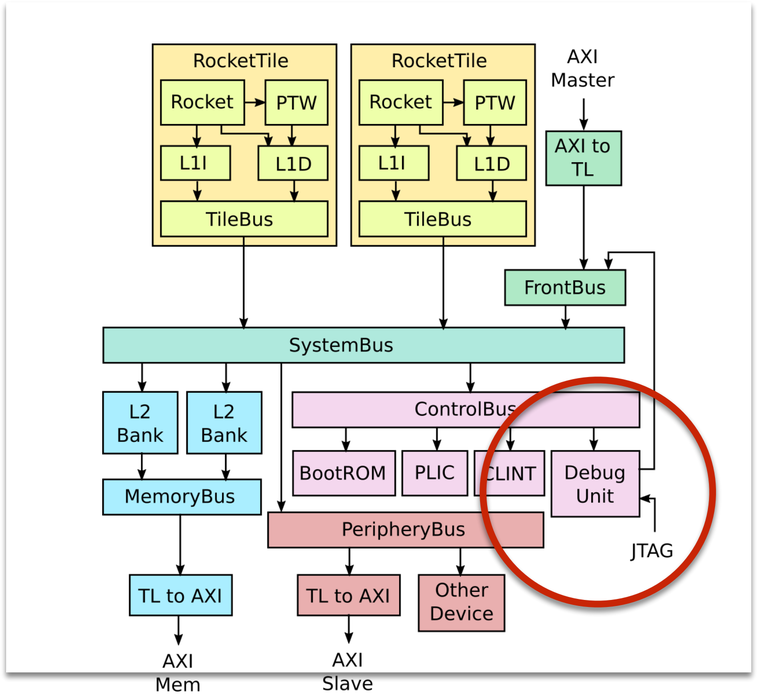
Don’t forget trace data!

The Definitive Guide to the ARM Cortex-M3 (2nd), Joseph Yiu, 2010 Figure 6.3

Make this relatively slow

RISCV has a similar system for TileLink





Make this relatively fast

Make this relatively slow

But where does Debug Module come from, Or

Who tests the tester?

Small hint: Make test chips…

Make a small test chip with just the debug module and put multiple copies

You don’t have to run it all full speed.

https://www.tsmc.com/english/dedicatedFoundry/services/cyberShuttle
https://gf.com/manufacturing-services/multi-project-wafer-program/
https://www.intel.com/content/www/us/en/foundry/intel-foundry-services/design-services.html

https://www.musesemi.com
https://themosisservice.com

While in real estate the refrain is “Location! Location! Location!” the comparable advice in IC design should be **“Testing! Testing! Testing!”**

For many chips, testing accounts for more effort than does design.

CMOS VLSI Design: A Circuits and Systems Perspective 4e, page 659

Neil Weste & David Harris

http://pages.hmc.edu/harris/cmosvlsi/4e/index.html

This talk only scratches the surface.

For a more complete introduction consider:

Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits
M. L. Bushnell and V.D. Agrawal, Kluwer
Academic Press, Boston 2000

Some Parting Advice :
Have a Simulation Regression Suite

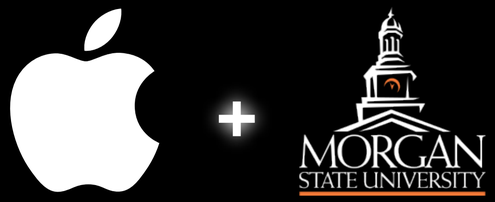
Have a simulation regression test suite for your ASIC. As the physical design evolves you'll need this to confirm functionality.

Bring up tests make for excellent regression tests. They are often fast and easier to simulate and produce a simple fixed result make them perfect for automated checks.

Some Parting Advice :
Reproducing Issues is the Key to Debug

Because measuring the internals of an ASIC is often indirect getting an issue to reproduce reliably is often the key to diagnosing the issue.

As you’re creating the test plan make sure you engineer ways to put your design into know fixed states so you can have known reference points to start debug from.



Questions?

nsi@apple.com