SoC 101:

a.k.a., "Everything you wanted to know about a computer but were afraid to ask"

Lecture 8: Ramp up and Debug a.k.a. "Lupulus: A Debug Story"

Prof. Adam Teman

EnICS Labs, Bar-Ilan University

23 June 2025



The Alexander Kofkin Faculty of Engineering Bar-Ilan University



Lecture Overview





Introduction to Lupulus





A Little Background

• The EnICS Labs SoC Platform:

- GENPRO • In the framework of the GenPro Consortium, the EnICS Labs were mandated with developing the "Israeli RISC-V Core"
- HAMSA-DI a superscalar version of the popular **RI5CY Core was the result of this project.**
- HAMSA-DI is delivered as part of PulpEnIX a full SoC Platform based on the Pulpino Platform from the PULP Project.
- PulpEnIX/HAMSA-DI had already been fabricated on three test chips in 65nm and 16nm FinFET.
- In 2022, we taped out "Bianca", a 16nm test chip, based on the **PulpEnIX SoC** platform.

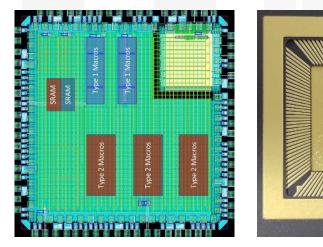


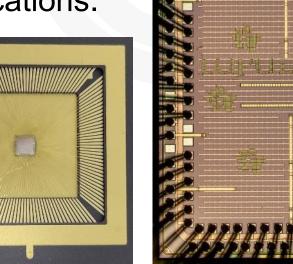
RISC-V

The Israeli RISC-V Consortium

Lupulus

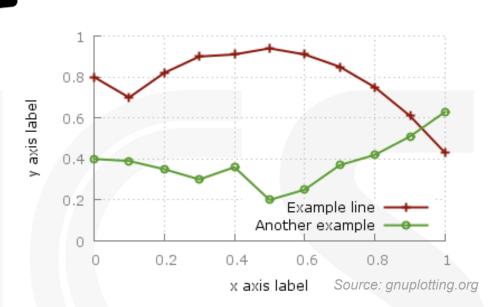
- As a follow up to Bianca, a second tapeout was sent for fabrication in 2023.
 - This tapeout included improvements to the test structures.
- Lupulus was based on the same Top Level design as Bianca, including:
 - PulpEnIX SoC Platform with HAMSA-DI host.
 - PULP-GCC compliant software stack with PulpEnIX bare-metal libraries.
 - Same 144-pin PGA package with almost identical pinout.
 - Compatible with same PCB with minimal modifications.
- The Lupulus test chip was delivered and bonded in October 2023.





What next?

- We have done some great R&D...
 Made something innovative.
- But how do we make sure that it actually works? How do we test it and make measurements to create those nice plots?



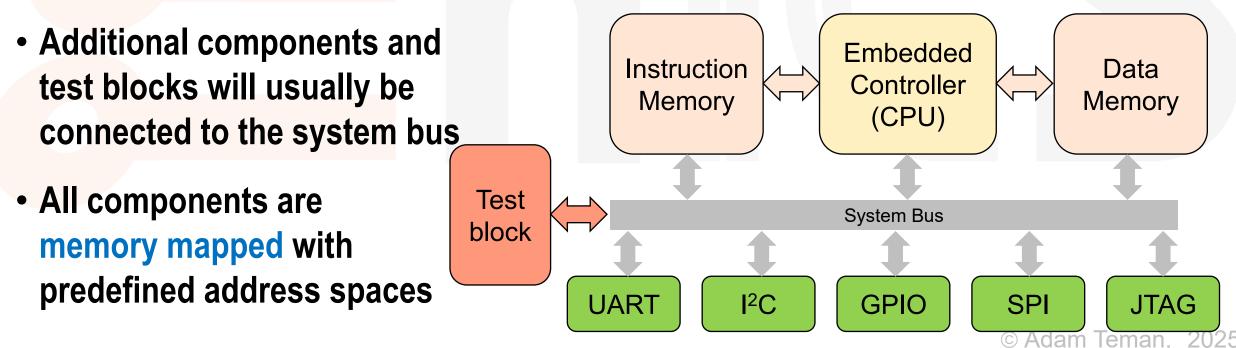
- The best way is usually to integrate your block with a control platform.
 An SoC!
- Let's remember what an SoC is and how it can help us before continuing.

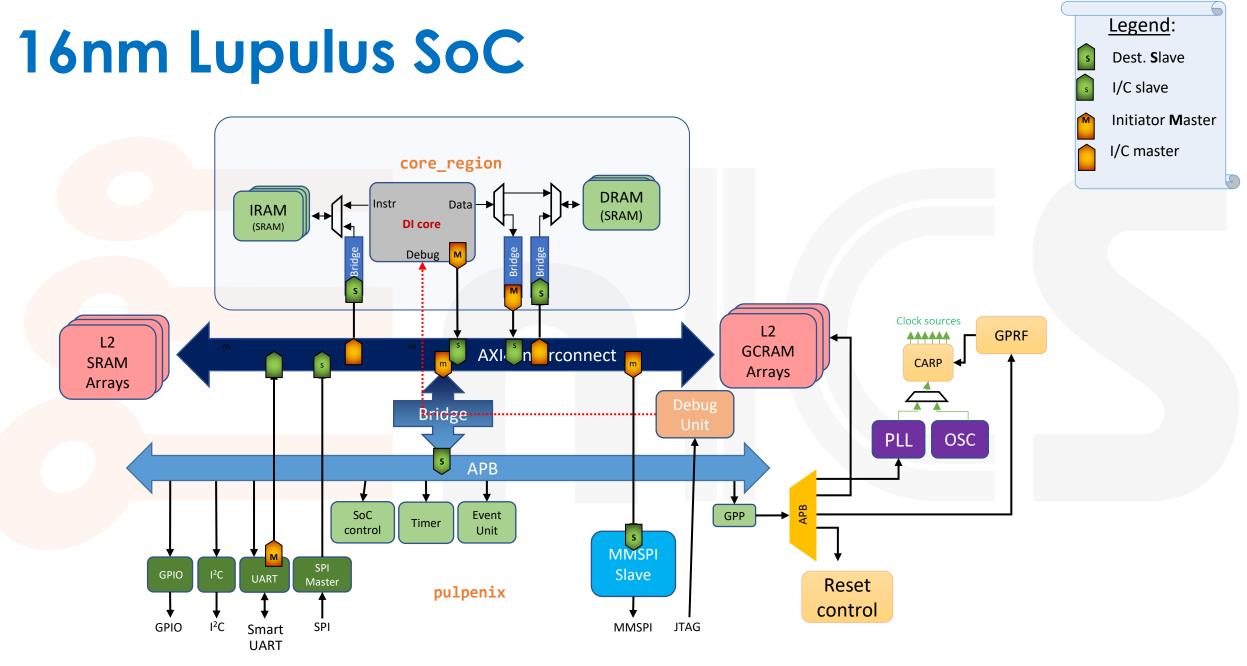
Back to our early SoC101 lectures

- A basic SoC will probably contain something like the following:
 - An embedded controller (a CPU) for running C-code

7

- Some tightly coupled memory for storing program code and data
- A bunch of peripherals for communicating with the outside world
- A system bus to send and receive data between the components





© Adam Teman, 2025



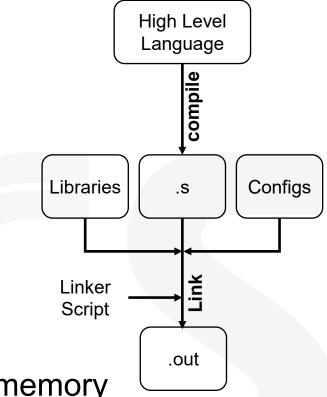
Booting Lupulus





Loading software on to the SoC

- The SoC is equipped with a software toolchain
 - Write programs in C code
 - Compile the programs to the ISA (e.g., RISC-V)
 - Use "bare metal" libraries for common operations (e.g., printf, gets)
 - Link with IRQs and boot code to configure SoC
 - Provide a linker script to direct binaries to instruction/data memory
- What actually happens when we hit the reset button?
 - A small number of hard coded (RTL) instructions are run
 - They load boot code ("bootloader") from an external source
 - The boot code loads the binaries to instruction and data memory
 - The processor jumps to the first address in the program (main ())

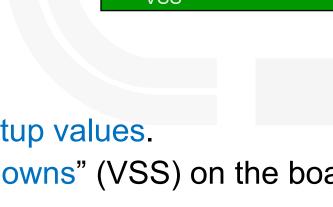


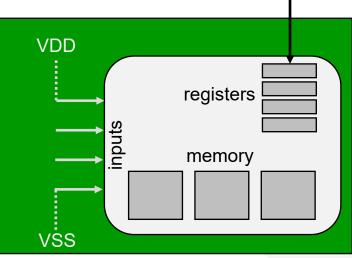
Configuring the Boot Sequence

• Remember that:

- The state of a system is set by its inputs and memory (including registers).
- Inputs come from pads that are connected to the board.
- Every flip-flop on the chip gets a reset value.
- Memory (SRAM) is garbage at start up.
- Configuration switches ("bootstraps")
 - The chip will have several inputs to configure startup values.
 - These are either tied to "pull ups" (VDD) or "pull downs" (VSS) on the board.
 - You can often change these by toggling a switch or using a jumper.

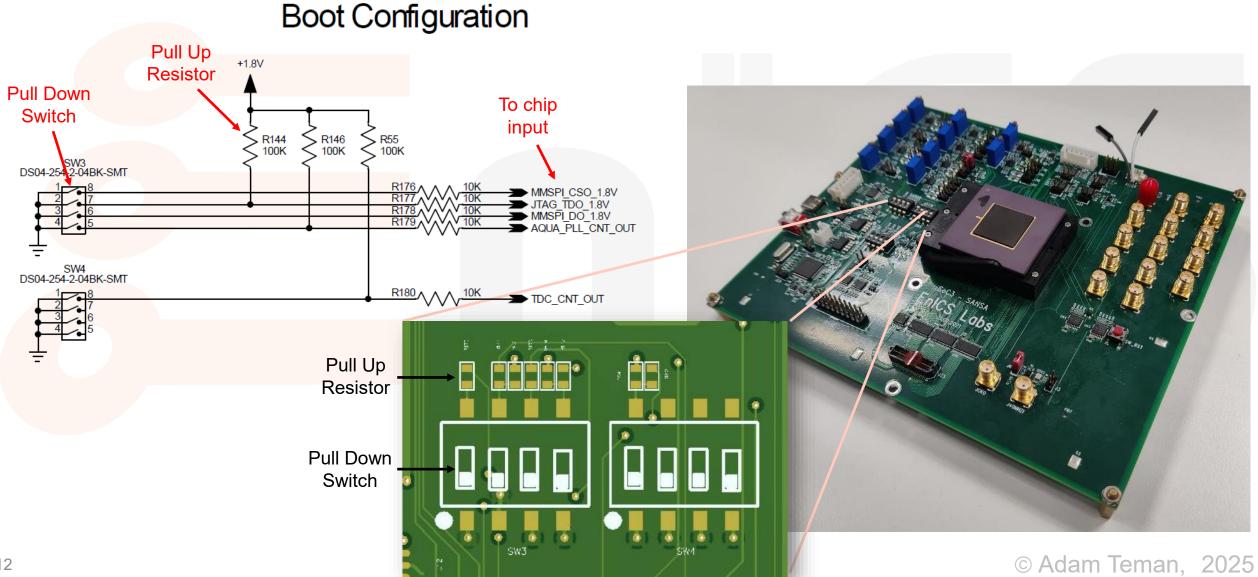






reset

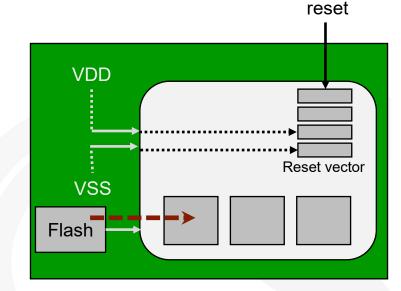
Example of PCB bootstraps



So let's go back to reset...

Reset Sequence

- When we hit reset, registers are set to default values.
- Some of the registers sample the pull up/pull down value from the board.
 - These can be configured through bootstraps!
- One of registers is the reset vector, which is the address of the program to run.
- This address may be mapped to an external device (e.g., Boot ROM or Flash).
- The short program is the bootloader that copies the program to SRAM.
- After the program is loaded, the CPU jumps to the instruction memory.



What happens in PulpEnIX

- Boot is (probably) the single-most "dangerous" part of your chip design
 - Udi's Drawer Analogy:



"Slam a drawer shut, but try to throw something into it before it closes"

- Therefore, always provide several alternatives.
- PulpEnIX has several methods for booting:
 - MMSPI:

As described before, Flash is memory mapped and reset vector points to it.

• JTAG:

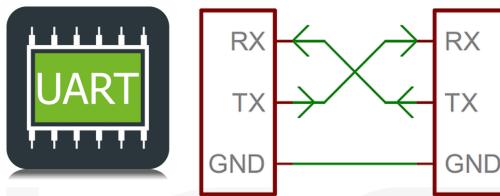
Utilize JTAG test ports to control the SoC with the OpenOCD standard.

• SmartUART + pyShell:

Backdoor into the SoC through the UART port. Then control the SoC through Interactive Python interface.

SmartUART

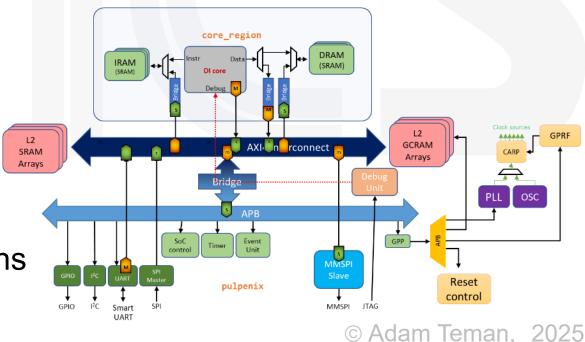
Reminder: UART



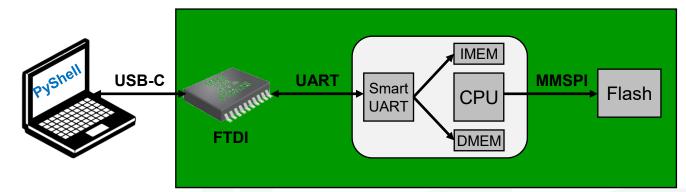
- UART is a very simple, two-wire (TX/RX) serial interface.
- PulpEnIX has a UART Slave port, i.e., only the CPU can initiate transactions.
- For example, bm_printf() sends characters over AXI/APB to a FIFO in the UART controller. The controller serializes the data and transmits it.
- This configuration does not provide a good debugging interface.

Introducing "SmartUART"

- Add an AXI Master port to the UART.
- Use UART to initiate transactions: read/write to all memory mapped locations



SmartUART



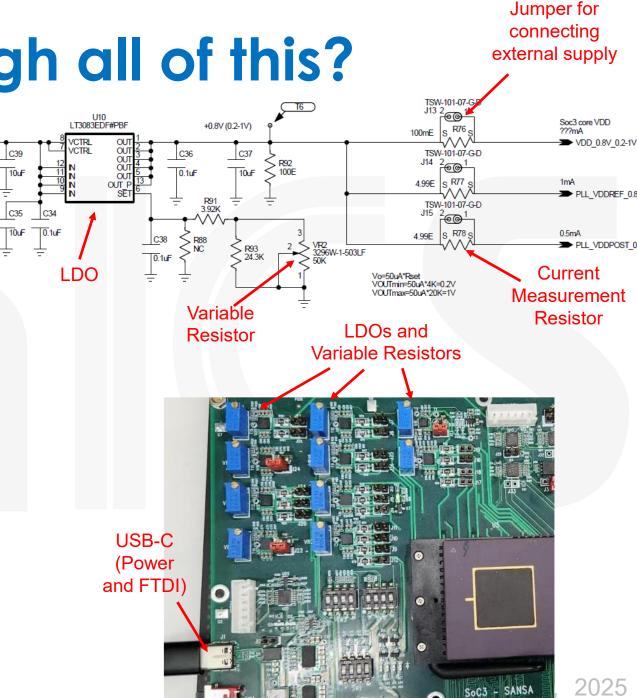
How does SmartUART work?

- Host PC is connected to FTDI chip, which translates USB-C into a bunch of signals, including UART, that are routed to chip.
- Reset tries to boot PulpEnIX through MMSPI by default.
- Host PC sends signal to chip (through FTDI) that stops the CPU.
- SmartUART sends write commands (Host PC through FTDI to AXI Master) to program registers and to load code into SRAM.
- Read commands from SmartUART bypass FIFO.
- How does SmartUART signal a read/write command?
 - UART transmits bytes (8 bits), usually representing ASCII chars.
 - Bit 7 (MSB) is unused in ASCII. This bit is utilized to signal transactions.
 - Python interface on host (PyShell) initiates transactions and polls for response.

Why did we go through all of this?

Voltage

- Well, our chips have just arrived.... What do we do now?
 - Set up our voltages!
 - Connect power supply and/or configure LDOs.
- Okay, Voltages all set. What now?
 - Run a program, of course!
- But it doesn't work...
 - Never does, unfortunately...
- So, we need to start debugging...
 - Thank God for SmartUART!





The Problems Start...





Hello, Who?

Lupulus was supposed to be a "plug-and-play" chip

- PulpEnIX platform taped out several times before.
- Package compatible with silicon-proven "Sansa" board
- Toplevel design (almost) equivalent to "Bianca" chip
- Really, just put the chip in the socket, run the "Hello, World!" program from Bianca, and no reason it won't work.
 (base) PS C:\Users\Lidor Geri\Documents\measurements_new_environment\Lup Compiling Application helloworld with link define=p, march=rv32imxpulpv3, opt=03, di=DI_ON, period

• So, why doesn't the chip react?

- Check voltages look fine.
- Check clock signal looks fine.
- Where is the program crashing? Looks like no UART communication.

• What now?

```
_apps\libs\sys_lib\src\rcg.c:10:3: warning: implicit declaration of function 'bm_printf' [-Wimplicit-function-dec
 bm_printf("starting init_pll\n");
 sw_apps\libs\sys_lib\src\rcg.c:13:58: warning: variable 'rd_data_cfg3' set but not used [-Wunused-but-set-variable]
 unsigned int rd_data_cfg0, rd_data_cfg1, rd_data_cfg2, rd_data_cfg3;
\sw_apps\libs\sys_lib\src\rcg.c: In function 'set_rand_pll_div':
 \sw_apps\libs\sys_lib\src\rcg.c:135:18: warning: implicit declaration of function 'rand' [-Wimplicit-function-declara
           = 1 + rand() \% 3;
  refdiv
 sw_apps\libs\sys_lib\src\rcg.c:132:41: warning: variable 'frac' set but not used [-Wunused-but-set-variable]
  int fbdiv, refdiv, posdiv2, posdiv1, frac, tmp, tmp ref;
\sw_apps\libs\sys_lib\src\rcg.c:132:8: warning: variable 'fbdiv' set but not used [-Wunused-but-set-variable]
  int fbdiv, refdiv, posdiv2, posdiv1, frac, tmp, tmp_ref;
type': 7, 'id': 67330065, 'description': b'Quad RS232-HS B', 'serial': b'B'}
ist of active serial ports:
 t:COM3 ; desc:USB Serial Port (COM3) ; hwid:USB VID:PID=0403:6011 SER=6
   Serial Port (COM3) OK
        ; desc:USB Serial Port (COM4) ; hwid:USB VID:PID=
  Serial Port (COM4) OK
rt:COM5 ; desc:USB Serial Port (COM5) ; hwid:USB VID:PID=0403:6011 SER=6
5B Serial Port (COM5) OK
rt:COM6 ; desc:USB Serial Port (COM6) ; hwid:USB VID:PID=0403:6011 SER=6
5B Serial Port (COM6) OK
 nd USB Serial Port at COM5
```

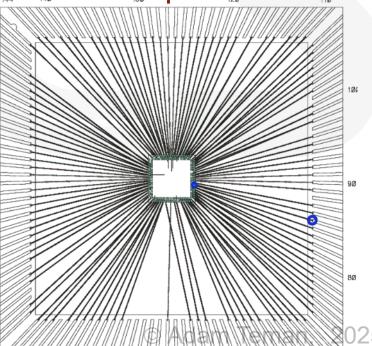
Was the chip bonded properly?

One possible point of failure is the chip pinout

- Chip passes LVS, so we know I/Os are compatible with design.
- But how did we actually check the bonding?
- Bonding is undoubtedly a potential single point of failure...
 - We have a spreadsheet mapping I/O pins to package pins to PCB pins

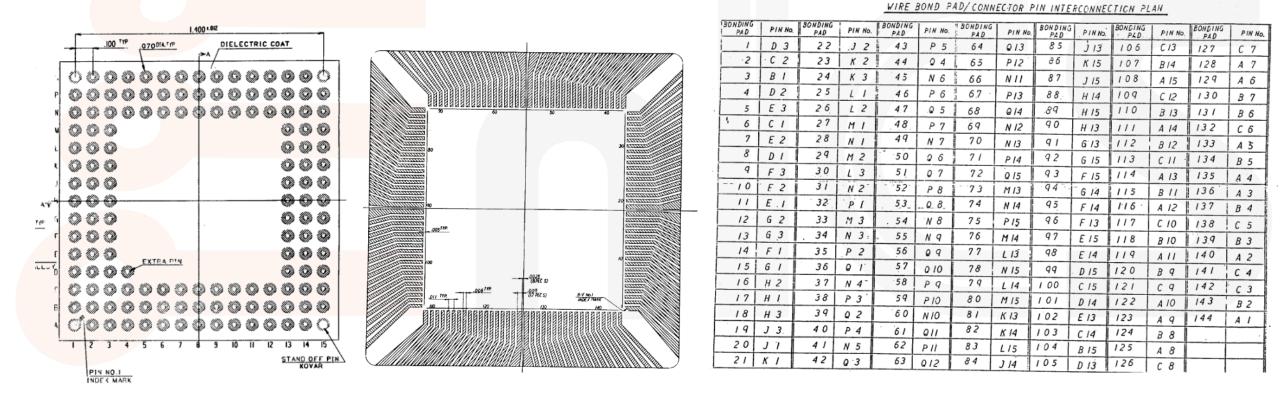
We also send a schematic of the connection between the chip and the package

	A	В	С	D
1	DIE PAD #	NET	PCK PIN #	PCB PIN #
2	1	PAD_GPIO_1	1	D3
3	2	PAD_MUL2B_PTL_VDD		
4	3	PAD_MUL2B_DML_VDD		
5	4	PAD_MAC_CMOS_VDD		
6	5	VDDIO	5	E3
7	6	VSS	6	C1
8	7	VDD	7	E2
9	8	VSS	8	D1
10	9	PAD_TDC_CNT_OUT	9	F3
11	10	PAD_AQUA_IN_22	10	F2
12	11	PAD_AQUA_IN_21	11	E1
13	12	PAD_AQUA_IN_11	12	G2
14	13	PAD_RST_N	13	G3
15	14	PAD_AQUA_PLL_CNT_OUT	14	F1
16	15	PAD_TOP_EDRAM_VDD_0	15	G1
17	16	PAD_AQUA_IN_9	16	H2
18	17	VDDIO	17	H1
19	18	VSS	18	H3



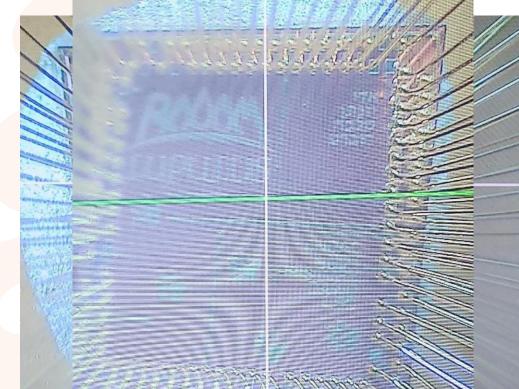
We also need the package spec...

Not that nice, considering the 144-pin PGA was designed in 1985...



So, let's try to look at the chip...

From our first look at the ship we thought it was rotated



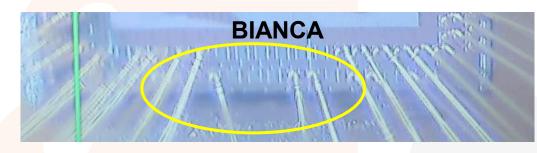
• But a closer look (and double check) showed that the orientation was correct.

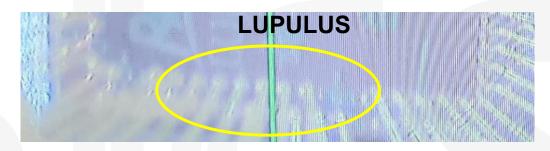
© Adam Teman, 2025

JPULUS

So, we started to compare the two chips

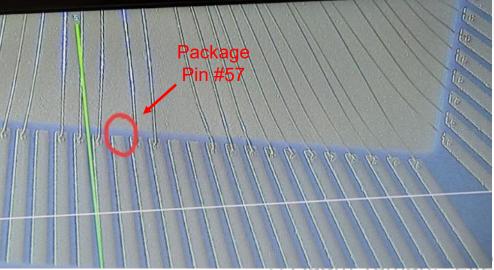
• Something was very strange on the bottom edge...





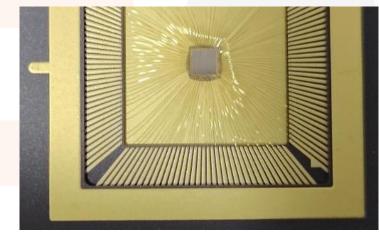
- It looked a little fishy...
- So we dug deeper...
 - Pin 57 isn't connected!
- What is it?
 - ...the clock!

47	PAD_AQUA_IN_3	47	Q5
48	PAD AQUA IN 4	48	P7
49	PAD_EXT_40M_CLK	57	Q10
50	PAD_JTAG_TMS	50	Q6
51	PAD_JTAG_TDO	51	Q7
52	PAD_JTAG_TCK	52	P8
53	VDDIO		
54	VSS	54	N8
55	PAD_JTAG_TRSTN	55	N9



We found the culprit. Life is good!

- Not connecting the clock is a good reason for nothing to work...
 - Basically, removing the chip from the socket gave the exact same response...
- We all breathed a sigh of relief and got ready for measurements.
- A quick fix at the bond house and everything will be good.



Not so fast!

• We got the exact same response from the chip!



.\sw_apps\libs\sys_lib\src\rcg.c:132:8: warning: variable 'fbdiv' set but not us int fbdiv, refdiv, posdiv2, posdiv1, frac, tmp, tmp_ref; ^~~~~~ 'type': 7, 'id': 67330065, 'description': b'Quad RS232-HS B', 'serial': b'B'}

ist of active serial ports:

port:COM3 ; desc:USB Serial Port (COM3) ; hwid:USB VID:PID=0403:6011 SER=6 USB Serial Port (COM3) OK port:COM4 ; desc:USB Serial Port (COM4) ; hwid:USB VID:PID=0403:6011 SER=6 USB Serial Port (COM4) OK port:COM5 ; desc:USB Serial Port (COM5) ; hwid:USB VID:PID=0403:6011 SER=6 USB Serial Port (COM5) OK port:COM5 ; desc:USB Serial Port (COM5) ; hwid:USB VID:PID=0403:6011 SER=6 USB Serial Port (COM5) OK

port:COM6 ; desc:USB Serial Port (COM6) ; hwid:USB VID:PID=0403:6011 SER=6 USB Serial Port (COM6) OK

ound USB Serial Port at COM5



The Problems Continue





Hello, Nothing

- So, we know that initially we had no clock.
 - Obviously, nothing worked.
- But we double-checked under the microscope and the clock is connected.
 - So how come it's as if the chip doesn't exist.
- Okay, there are some things that are different now...
 - Playing with the bootstraps, gets some activity from the chip...
 - For example, we can see that it is trying to fetch code from the MMSPI...
- But the UART doesn't work, and UART is really, really simple
 - It's such a slow interface that lowering the frequency wouldn't help...
 - Furthermore, we've used this IP on many chips, so it's not a logic bug...
 - If it's not a bug and not the frequency could it be a hold violation?

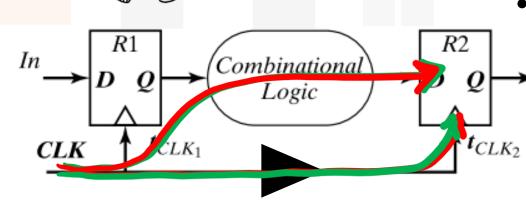


Back to our STA basics

- Max Delay (Setup)
 - The data arrives at the capture reg later than the following clock edge:

$$T + \delta_{\text{skew}} > t_{\text{cq}} + t_{\text{logic}} + t_{\text{setup}} + \delta_{\text{margin}}$$

- Possible reasons:
 - Slow corner (long delay)
 - High frequency
 - Negative skew
- Solution?
 - Lower the clock frequency



- Min Delay (Hold)
 - The data arrives at the capture reg before the same clock edge:

 $t_{\rm cq} + t_{\rm logic} - \delta_{\rm margin} > t_{\rm hold} + \delta_{\rm skew}$

- Possible reasons:
 - Fast corner (short delay)
 - Positive skew

Solution?

- Throw away the chip?
- Maybe we can make the chip slower???

How can we make the chip slower?

• Remember, PVT = Process, Voltage, Temperature

- Process... well, we probably have a typical chip. We could try several, but...
- Temperature... yeah, we have an oven. We could heat it up (or cool it down...)
- Voltage... hey, that's pretty easy, let's just lower the voltage!
- Slightly lowering VDD and something starts happening!
 - We perform write to several addresses from the UART and can successfully read them back!
 - But when we write to certain addresses, we read back "deadbeef"
- What in the world is "deadbeef"?
 - It's a hexa number that tells us that something isn't right...
 - PulpEnIX responds with deadbeef when you access an illegal address...
 - That means... that... something is working... READ is working...

Maybe we changed the memory map?

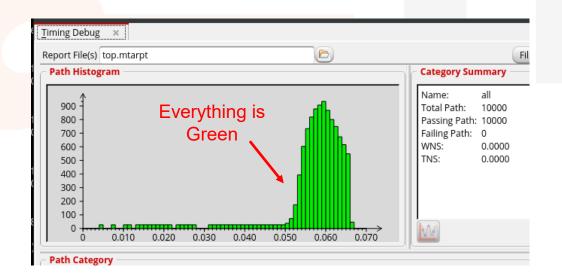
• Remember our good old "memory map"?

- When we design a chip, we give each device a region of addresses.
- These are hard coded into the chip the synthesized hardware checks the address according to the defined memory map.
- This is another *potential point of failure*, since we define it manually (in an Excel sheet...)
- But, no...
 - The memory map is the same as in Bianca.
 - Gate-level simulations of booting through UART show everything is okay.

	File Edit View Search Terminal Help
	uart_tb.sv: Remote Py Command: W 00100540 1a30422c
	uart_tb.sv: Remote Py Command: W 00100544 1a3040c8
	uart_tb.sv: Remote Py Command: W 00100548 1a304254
	uart_tb.sv: Remote Py Command: W 0010054c 1a304200 uart tb.sv: Remote Py Command: W 00100550 1a304218
•	uart tb.sv: Remote Py Command: W 00100554 00100534
	uart tb.sv: Remote Py Command: W 1a107008 00000000
	uart tb.sv: Remote Py Command: W 1a3010c4 00000001
	uart tb.sv: Remote Py Command: C
	uart_tb.sv: Remote Python changed TB clk_period_ns from 25.00 to 3.33
	uart_tb.sv: Remote Py Command: U
	uart_tb.sv: Remote Python changed UART clk_div_cntr value to 14 (decimal)
	uart_tb.sv: Remote Python changed UART ns per bit_actual from 150.00 to 50.00
	uart_tb.sv: Remote Py Command: S
	uart_tb.sv: Remote Python request to simulate for 250000 (decimal) time units
	uart tb.sv: Remote Py Command: R 00000000
	Hello
	/ V\
	<<
	<)()

So, it really looks like a hold violation...

- Let's further reduce the voltage...
 - More addresses start to work.
- Even more...
 - At 0.67V we get a "Hello, World!"
- But it can't be... we checked hold before tapeout...





h: 19		Worst UART path >500ps sl
be:	Hold Check, reg->reg, 8 segments	View: FUNC_FF_HiV_ZeT_hold_Cbest_CCbest
ck:	0.0510 (req. sme: 0.0760, arr. time: 0.1270)	Skew: 0.000000 (Incr Delay: 0.0)
PR:	0.0000 C	PPR Common Point:
rt:	I_pulpenix_top_peripherals_i_apb_uart_sv_i_su (clocked by PNX_CLK leading, latency: 0.0000)	1_master_ascii_cmd_done_s_reg/CP (DFCNQD1BWP16P90CPDLVT)
d:	I_pulpenix_top_peripherals_i_apb_uart_sv_i_co (clocked by PNX_CLK leading, latency: 0.0000)	pnv_ascii_rx_cmd_addr_data_val_reg_37/D (DFCNQD0BWP16P90CPDLVT)
ek Ca	Joulation	

Slack Calculation

Path

Тур Slad CPP

Star

End

Data Path Capture Clock Path SDC Timing Interpretation Schemati

Data Dela

Name	Arc	Cell	Delay	Sum	Status	Load	Slew	Incr Delay
I_pulpenix_top_peripherals_i_apb_uart_sv	CP->Q	DFCNQD1B	0.067	0.067			0.086	-0.001
I_pulpenix_top_peripherals_i_apb_uart_sv			0.004	0.071		0.024	0.094	-0.002
GNS_i_671173	B1->ZN	INR2D0BW	0.022	0.093			0.009	0.000
n_230627			0.000	0.093		0.001	0.009	0.000
INVS_route_I_FE_PHC110322_n_230627	I->Z	DEL025D1	0.024	0.117			0.005	0.000
INDEX south a FF DUNITION A 200607			0.000	0.117		0.001	0.005	0.000

30

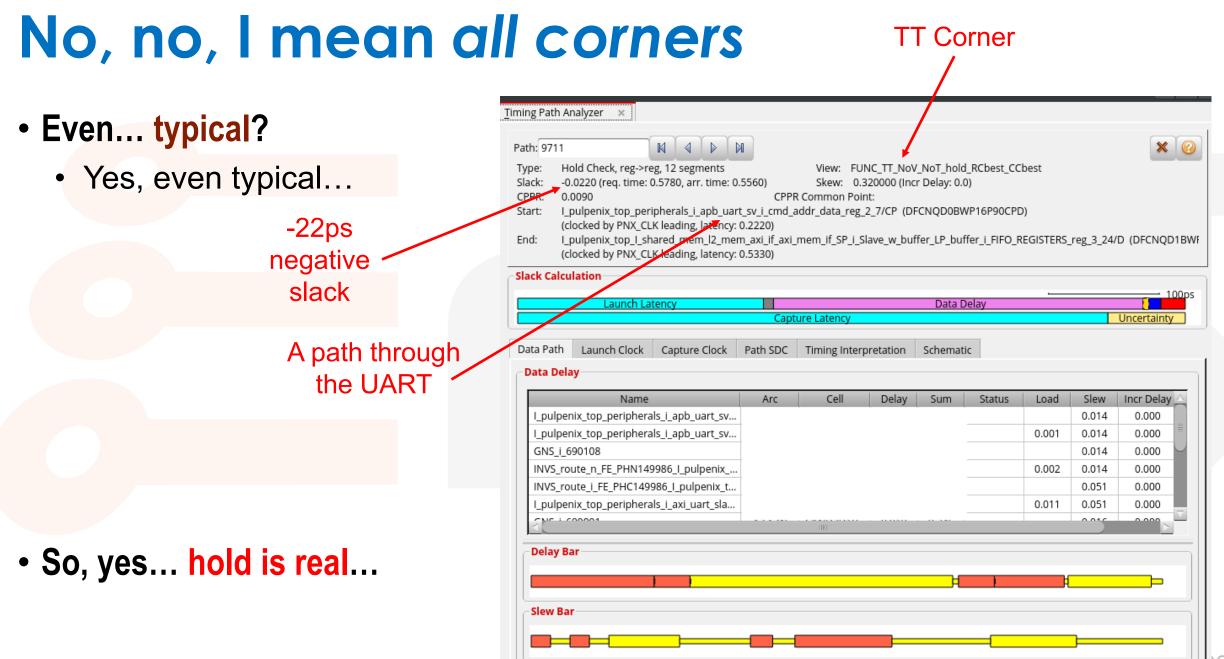
Are you sure you checked all corners?

- Well, didn't we just say that ...?
 - Worst case setup is in the Slow corner.
 - ...and worst-case hold is in the Fast (best case) corner.
- Yeah, but at nanoscaled technologies, things get weird
 - During signoff, you have to check hold at all corners.
 - ...and there are a lot of them!
 - Remember, RCbest, RCworst, etc.



R	Path	Cleck	RegTime	Slack	Logic Level	Sum of Wire Length	Startpain Fin	
	99870	PNI_CLR(hadrig)=P	1.112	-0.179	34	1,296.31	(julperia.top.) (hand.mem.) (2.edram.rms.mma.to.regRP	1, page tik top 1 shared meet 1.02 edtare gran
	0.048	Plat. CLEDwadergh iff	6.540	4,176	0	100	(polyaris top pergharat) (aplicuart to 117 A add. data reg. 6	1.5CP (polyamic trap core region) data memuaic P a
	9972	PNI_CLR(hading) -P.,	1.113	-0.176	33	106 294	Lpulperis.top.Lphared.mem.U.D.edram.m.e.medic.reg/CP	(pulperie.top.).shared.mem.).Q.odram.goran
	9973	Phil_CLR(heating)=P.	0.941	-0.176	29	27.4.872	(pulperis/top.).phared_mem.).02.edram.mac/mac/cog/07	(pulperischip.).shared.mam.).02.edram.grouw
	9874	PNI_CXEssing-P.	9.527	-0.176	31	982,391	(palperis, top.), shared, mem, J.D., edram, max, maptiped, rag(CP	(pulperis_top)_shared_mem_(.02_edram_goram
	9975	PNX_CLR/exdings -P	1.121	-0.179	52	1214.187	Lpapera, top. Lphand, men. L12, editam. me, mack, regiCP	1,pulperis.trp.1,shared.mem.1.02.edram.gotan
	9875	Phil_CLR(beatings-P.,	5.525	-0.178	3.5	1336.445	(palperia.top.).shared_mem.j.D.edram_max_max_b_regR7	(_pulperis_top_)_shared_mem_12_edram_grout
	9977	PNI, C.F.Badrigh-P.,	0.943	-0.176	21	1177.162	Lpdperis.top.1.shared.mem.1.02.ed/art.rma.is.rep07	(pulperix.top.(shared.mem.).Q.oham.goram
	9979	PNR_CIR[leading] - P	1.136	-0.176	37	1297.809	(paperis,top.),shared_mem_U2,edra_rma_ma_is_regi27	(pulperia.top.).shared.mem.).22.edram.goram
	3975	Phil_CLR(heading)-P.,	0.941	-6.176	28	1121.671	(palperia.top.) shared_mem.).(2, educe_mma.to_eq/CP	(pulpania top.) shared mem 1.02 adram goran
		PRO CLASSING STOR				104.541	(poperts top perpherats) and cart of 1 cmd and data reg fi	(polyana, tag.), provided and the second sec
								LACE Conferences and experiences of an index of the
	9962	PRI_CLR/kadings-P.,	0.943	-0.176	10	1246.121	Lpalperis.top.Lshared.mem.L12.edram.rma.tmack.rep07	Path 2081 (rep srept
	19403	PMI, Ci Ribeatings - P.,	1.004	-0.176	29	2134.540	(pulperia.top.),shared_mem.(.0,edram_rmacis_regi0)	File: held all connect.mtarpt Start I pulperix top peripherals I apb uset so I cred addr i
	2264	PNI, C.Kihadrag-P.,	0.941	-0.176	28	1891.327	Lpdperis.top.1.shared_mem.1.0.edram_mackmedia.regRP	End i pulpanix top aat interconnect i aat node i REQ 85.0
	1985	PNR_C.R/kading-P	6.943	-0.176	23	1525.321	(pulperia.top.) shared mem.).(2.edram.rma.mma.ik.repRP	1 pulperix, top (shared, mem,) (2, edram, grram
	1	PER CLICKARD					(Judgeres tray perpheral Logit, car to Love Judit, data reg.)	(). () () () () () () () () () () () () ()
	9967	PNI_CURIhading-P	0.943	-6.178	24	1198.362	(pulperia.)sp.).shared.mem.).02.edram.mea.mea.ja.regi07	(pulperix.top.).shared.mem.(.0.idram.gorph
	9983	PNI, C.Rihadogi-P.,	1.113	-0.176	38	1267.368	(pdperis.)op.(shared.men.).0.edram./majmask.rep07	(pulperis/to).(shared.mem.).0.edram.goram
	1983	PNI_CLR(kadings-P.,	0.943	-0.176	23	1418.0	Lpdperis.top.Lshared.mem.L32.edram.mac.mack.regRP	(judperin, top.), shared, mem, 132, idram, goran
	1990	Phil, C.Kibadreg-P.,	0.943	-0.176	19	1288.032	(pulperis.top.),shared.mem.).Q.edram.rma.mae.b.regiO7	(julgeris, trip.), shared, mem.). (2, edram, grran
	9991	Phil_CiR[hading)=P.,	1.118	-0.176	31	1290-439	(pulperia.top.).phared_mem.(.Q.edram_mma_in_regiCP	(pulperis.trp.).shared_mem.).0_edram.grram
	9993	PNR, C.R(hadreg) = P.,	0.941	-0.176	21	1236.979	(pdperis.)sp.j.shared_mem.).(2,sdram_mme_ik_regiOP	(pulperix,top.),threat,mem.).12, adram.goraw
	0993	PNI, Citilitatingh-P.,	0.943	-0.176	24	1166.281	(polperis,top,),shared_mem_U2,edram_mma_k_regiCP	(pdperis.trp.).share(.nem,1.0,etram.gram
	9994	PNI, CLK(bading) - P	1.121	-0.179	31	1376.183	Lpulperia.top.j.phared_mem.j.Q.edram_mme_ta_regiOP	(pulperie.top.).shared.mem.).02.edram.goran
	9995	PNI, C.Nitesting-P.,	1.522	-0.179	34	1540.338	(julperia.top.).shared.mem.).02.edram.maa.maa.to.ptp07	(_pulperis_trp_)_share(_mem_)_2_edum_grave
	0995	PNI, C.R(hadrog)-P.,	0.941	-0.176	28	1704.193	Lpulperis.tsp.1.shared_mem.1.02.edram_mma_ts_regRP	(pdpericitip.) shared men.) (2, edram gram
	1000	Pitt (Littleading) #					Louiperin top perpherais Lapit said to Licens and in cred at	the data will reg 2007 (pulperior top, core, region) data ment, and if a

But UART paths are within the uncertainty margins



<u>Same 10025</u>



But, we're not done yet...





Ready, set, go!

- Clock is connected.
- UART works when voltage is lowered.
- CPU works great.
- What about our memories?
- They don't do anything, of course...
 - Whatever we do, they always read out '1'
- Ahhhhrrrggghhhhh!

10w 490
fffffffffffffffffffffffffffff
row 497 : fffffffffffffffffffffffffffffffffff
row 498 : fffffffffffffffffffffffffffffffffff
row 499 : fffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffff
row 500 : fffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffff
row 501 : fffffffffffffffffffffffffffffffffff
row 502 : fffffffffffffffffffffffffffffffffff
row 503 : fffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
row 504 : fffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffff
row 505 : fffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffff
row 586 : fffffffffffffffffffffffffffffffffff
rrrrrrrrrrrrrrrrrrrrrrrrrrr
row 507 : fffffffffffffffffffffffffffffffffff

row 598 : fffffffffffffffffffffffffffffffffff

ffffffffffffffffffffffffffffffffffffff
PON 510 : TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
Direct test summary: Failed bits = 524288
Failed Dits = 5/4/28
Refresh period = 10 us
Neiresii periou - 10 us



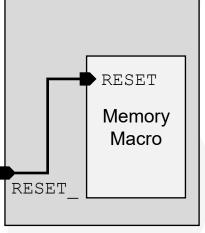
Direct test summary: Failed bits = 524288 Tested bits = 524288 Refresh period = 10 us

© Adam Teman, 2025



- After a few days of searching for a needle in a haystack we found the problem.
- The memory macro has a **RESET** input
 - This disconnects the sense amplifiers and outputs a constant '1'...
- But the SoC has a RESET signal
 - So, the minute the SoC releases reset, the macro is reset...
- How did this happen?
 - Another potential point of failure the Behavioral Model.
 - Logic simulations are run with a Verilog model of the IP that is written by hand.
 - The behavioral model was not equivalent to the analog IP.
- Solution
 - Always run (at least basic) mixed-signal verification!

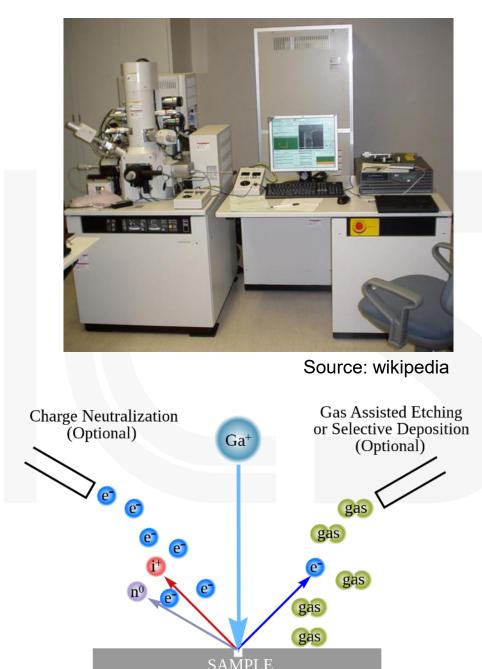
<pre>module memory(, RESET_);</pre>
always@(posedge clk or negedge RESET_)



So, the chip is dead...

Not so fast!

- We have one more rabbit up our hat...
- The FIB!
 - A Focused Ion Beam is a machine that enables post fabrication circuit editing.
 - At low currents, a FIB enables high resolution imaging (similar to a SEM).
 - At high currents, the FIB can drill down (etch) into the chip with sub micrometer precision.
 - It also can be used to deposit a conductive metal layer to make a new connection.
- Could we fix the memory with a FIB?
 - Luckily, Yoav is an expert!

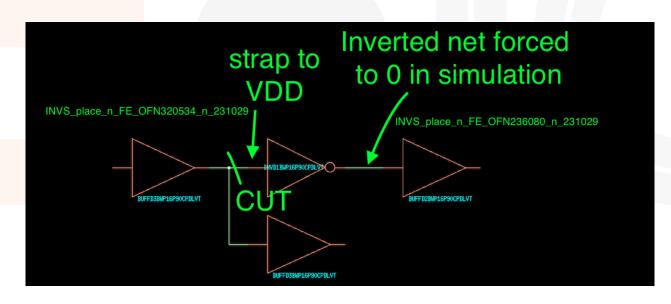


2025

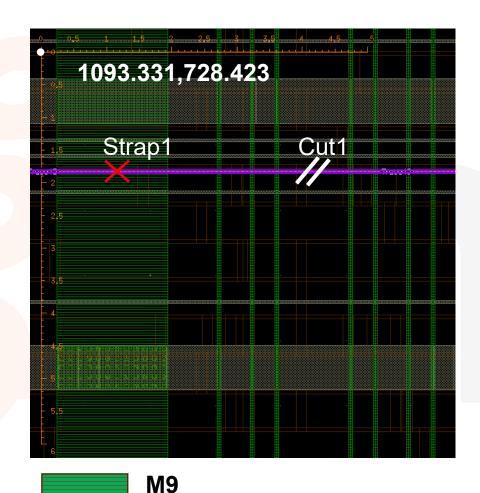
© Adam leman.

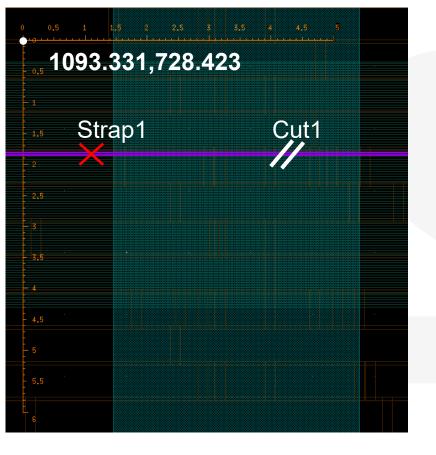
Step 1: Simulate the fix

- Find a reset net that is close to the macro.
- Connect it to VDD.
- Simulate it and see that all is okay.



Step 2: Find a place to drill







M10 (Over interested line)

M11



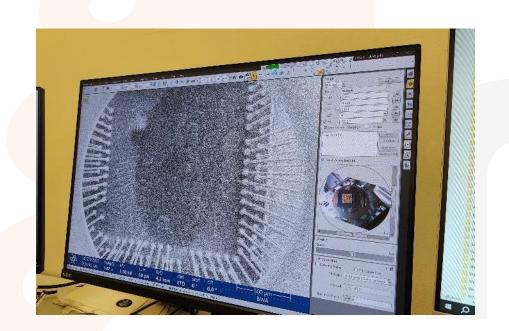
M8

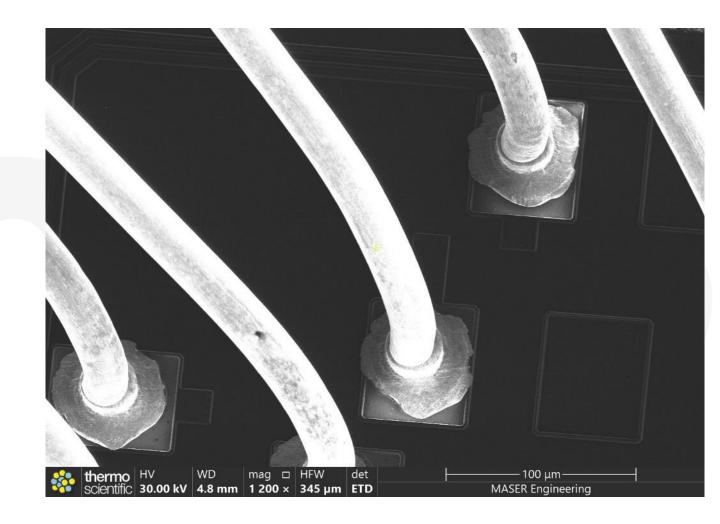


© Adam Teman, 2025

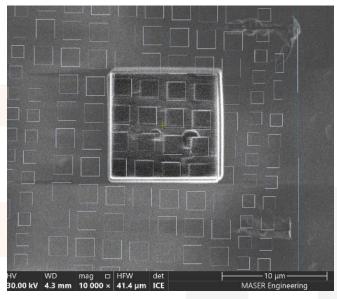
Step 3: Go to the FIB

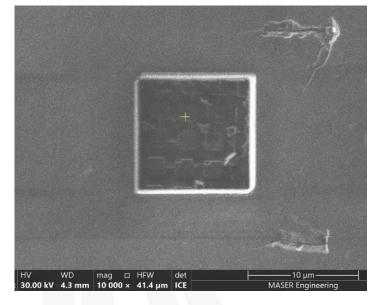
• It's all or nothing now....

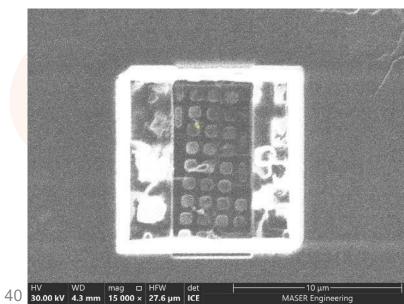


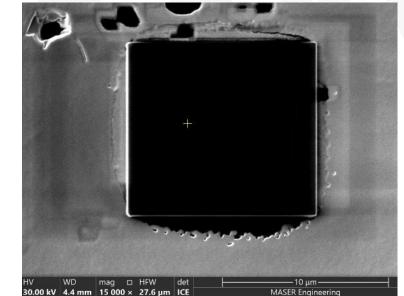


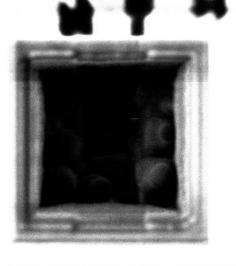
Focusing in on the ROI







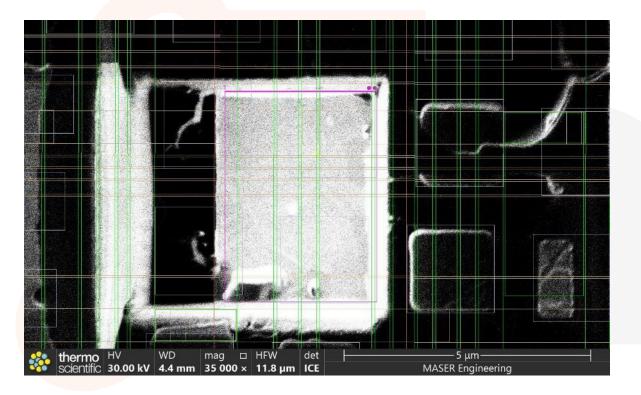


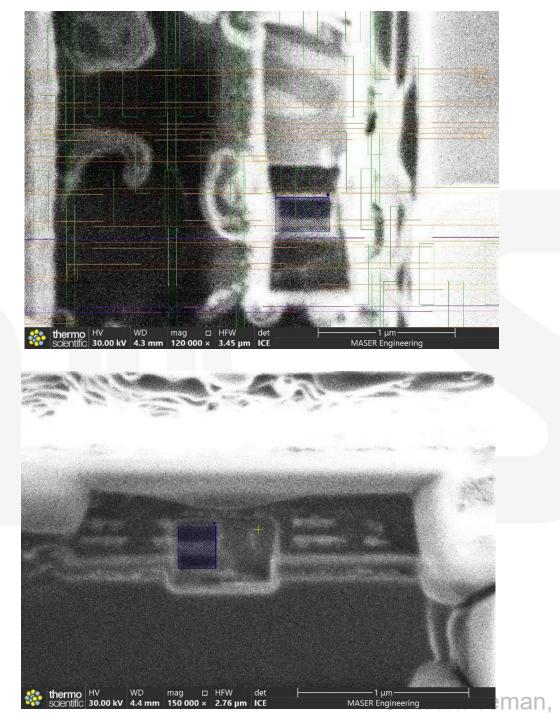


WD mag ⊡ HFW det

30.00 kV 4.3 mm 15 000 × 27.6 µm ICE

Going deeper





And here goes nothing...



Conclusions

Adhere to Signoff guidelines

• Signoff checklists are "written in blood". Don't take them for granted.

Identify potential points of failure

 Anything that is not automated and independently verified is a bug waiting to happen.

Patience and Persistence

- Nothing ever works on the first shot.
- Don't give up. There's almost always another workaround or fix. You just need to find it.

Know all the fine details

- Nothing "magic" happens in engineering.
- Debugging is like a mystery. You need all the clues to reach the solution.

Acknowledgements

- This adventure required a lot of help to get to the finish line.
- Thanks to the EnICS Labs, RAAAM Technologies, Eurofins MASER, Beckermus Technologies for their participation in the debugging process.
- Special thanks to Udi Kra, Yoav Weitzman, Yonatan Shoshan and Christoph Mueller.