Designing Multicore Microcontrollers

with some help from my friends

By Chip Gracey of Parallax, Inc.

Me after Propeller 1 was finished, working on Propeller 2 – six years ago.

Easy Fast Cheap Online Traffic School
4901 Morena Blvd. Suite #214-C
San Diego, CA 92117
http://www.easyfastcheaponlinetrafficschool.com/
1-800-554-2718

Dear Charles Gracey,
Enclosed are two copies of your Traffic School Certificate of Completion.
Court Copy: It is your responsibility to sign, date and get the Certificate of Completion to the court before your due date.
Student Copy: Enclosed is your Traffic School Certificate of Completion. Keep your student copy of your certificate for at least three years for your records.

Easy Fast Cheap Online Traffic School accepts no responsibility for lost or misrouted mail from the US Postal Service, Overnight Delivery Services, or the Court. All services are subject to the limitations and requirements of the United States Postal Service, Overnight Delivery Services and the Court. We will not accept any responsibility for lost or mistakenly routed mail.

Thank you for your business. Please buckle up and drive safely.

Kindest Regards,
Easy Fast Cheap Online Traffic School

Traffic School Proof of Completion

Student Copy -- Keep this for your records
HS No: 2042 - Easy Fast Cheap Traffic School

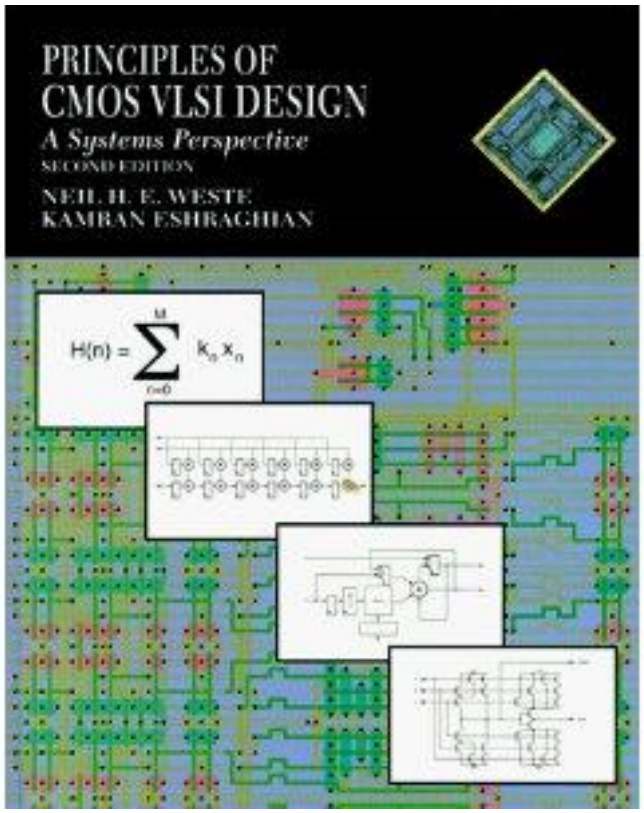
County & Court: Yuba, Yuba / Marysville Co
Student Name: Charles Gracey
Address: 12967 State HWY 99E
Red Bluff, CA 96080
Date of Birth: /1967
Case/Docket Number:
Driver License:
Completion Date: 4/8/2012
Due Date: 4/10/2012

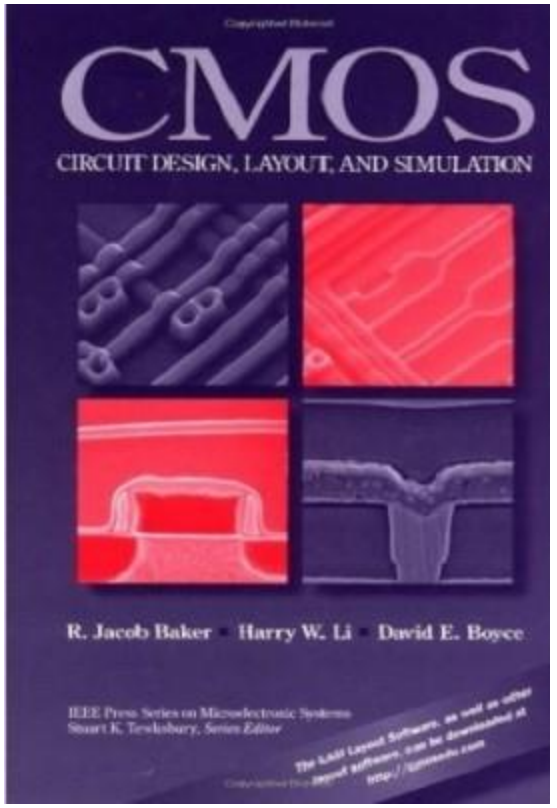
Charles Gracey, has completed the Easy Fast Cheap Traffic School Program requirements according to the applicable rules and regulations.

4/8/2012
Student Signature Date

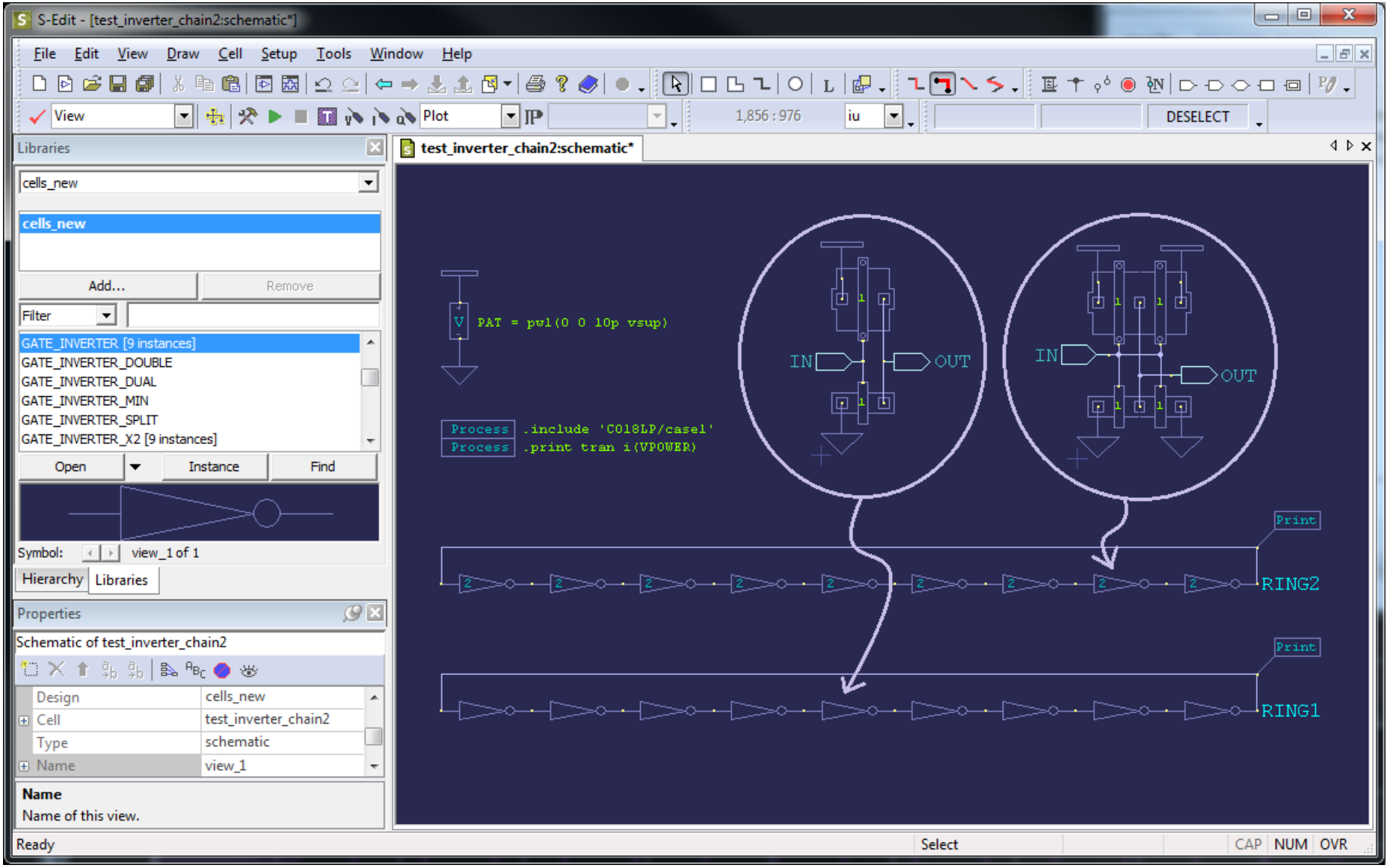
M. Keller 4/8/2012
Issued By M. Keller Date

You don’t need fancy accreditations to design your own chips.

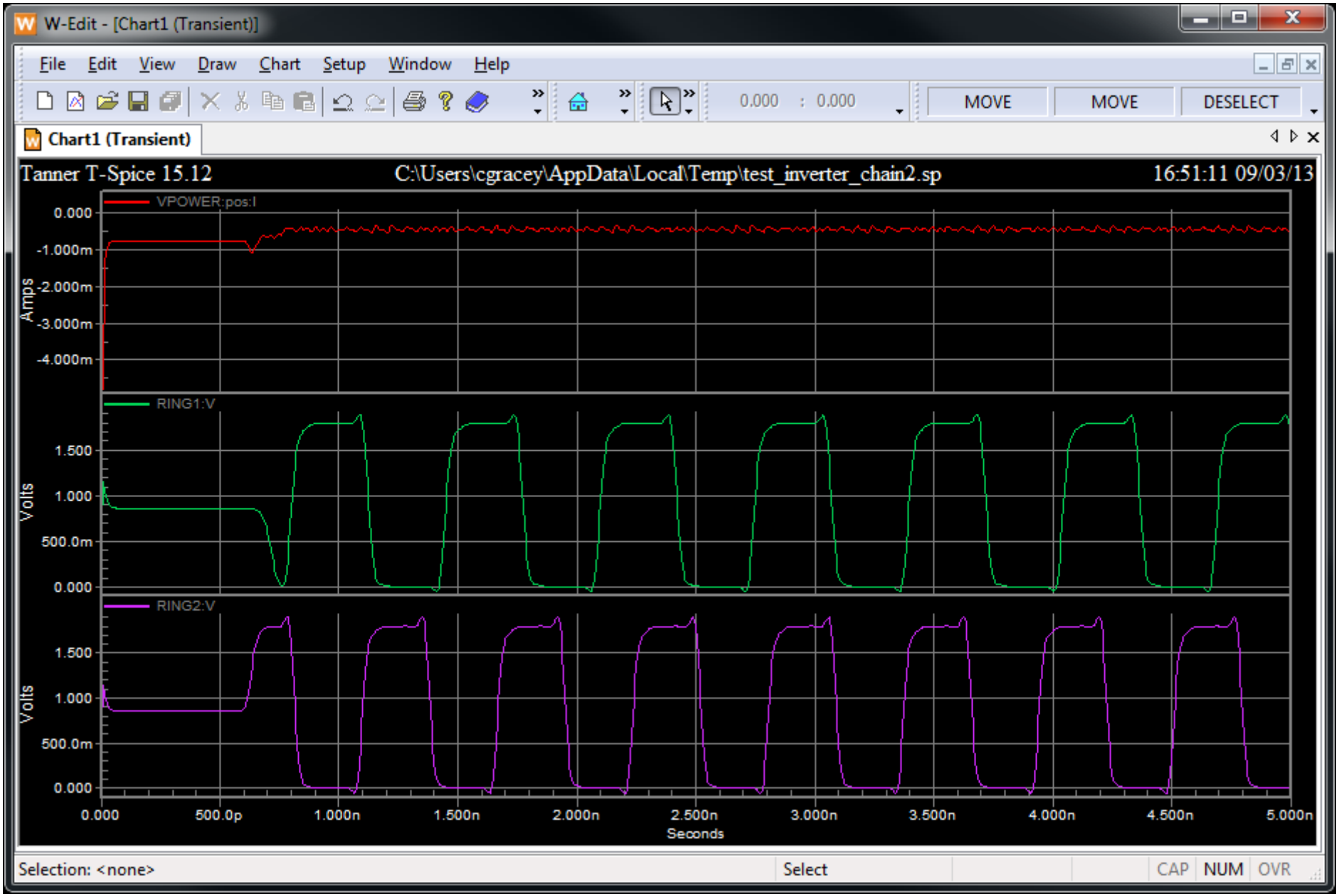




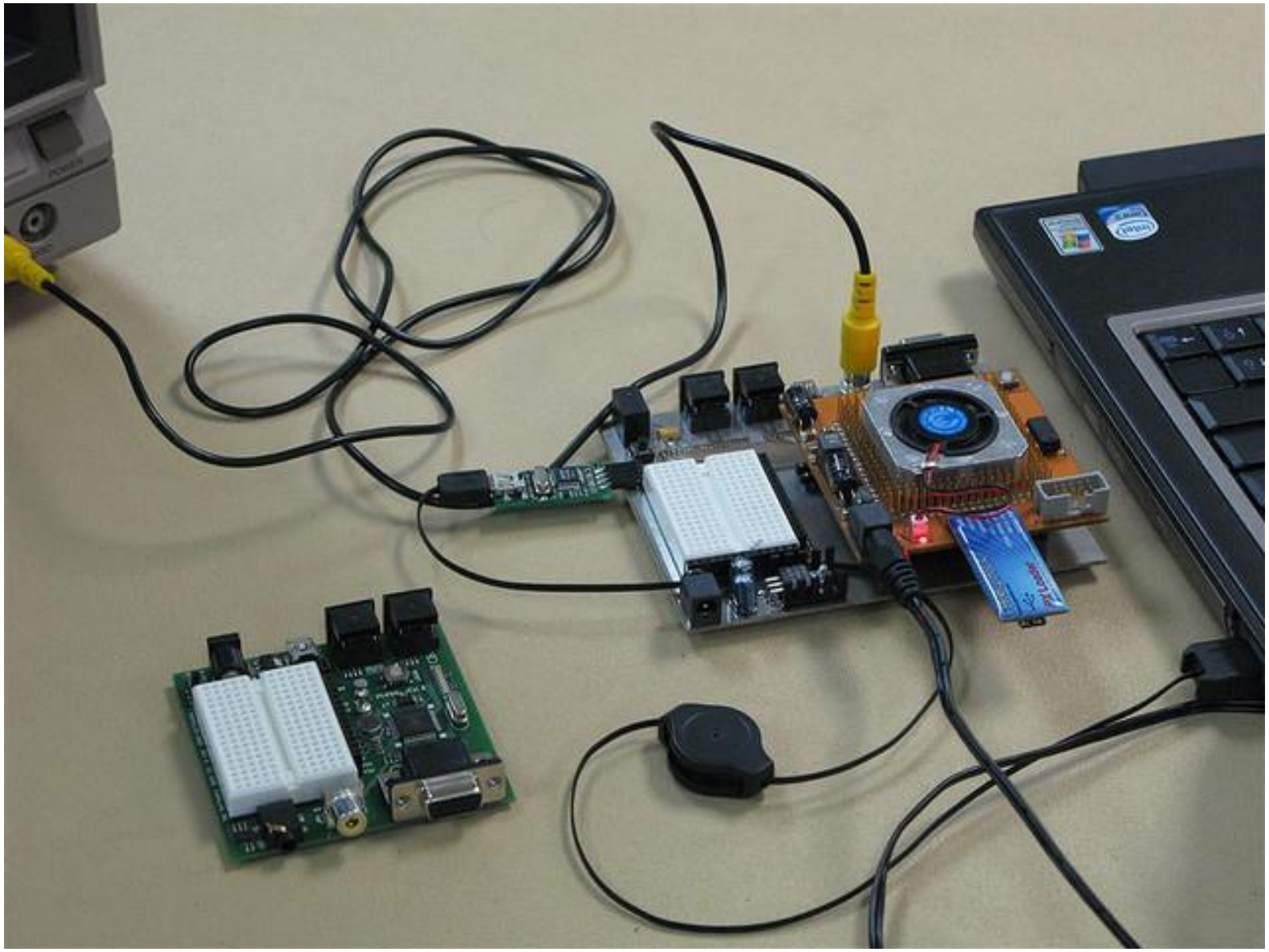
Everything you need to know about full-custom chip design is in these two books.



Full-custom design is nitty-gritty, lots of fun. You control EVERYTHING.

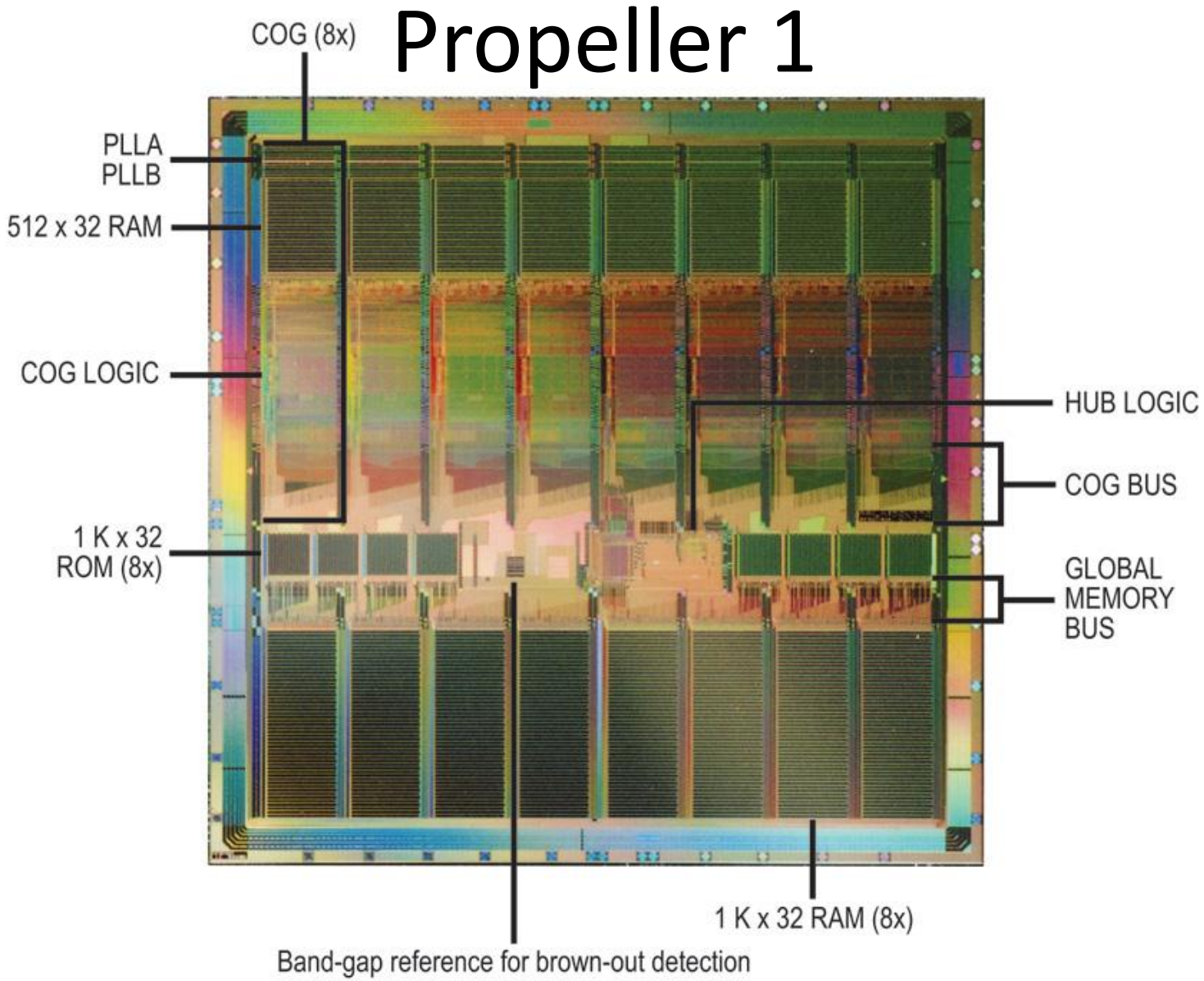


RING2 with the 2X shared-drains inverter is FASTER. It has lower parasitic capacitance per drive strength. Neat-O!



An Altera FPGA board I designed to develop Propeller 2 on sits next to a Propeller 1 Demo Board.

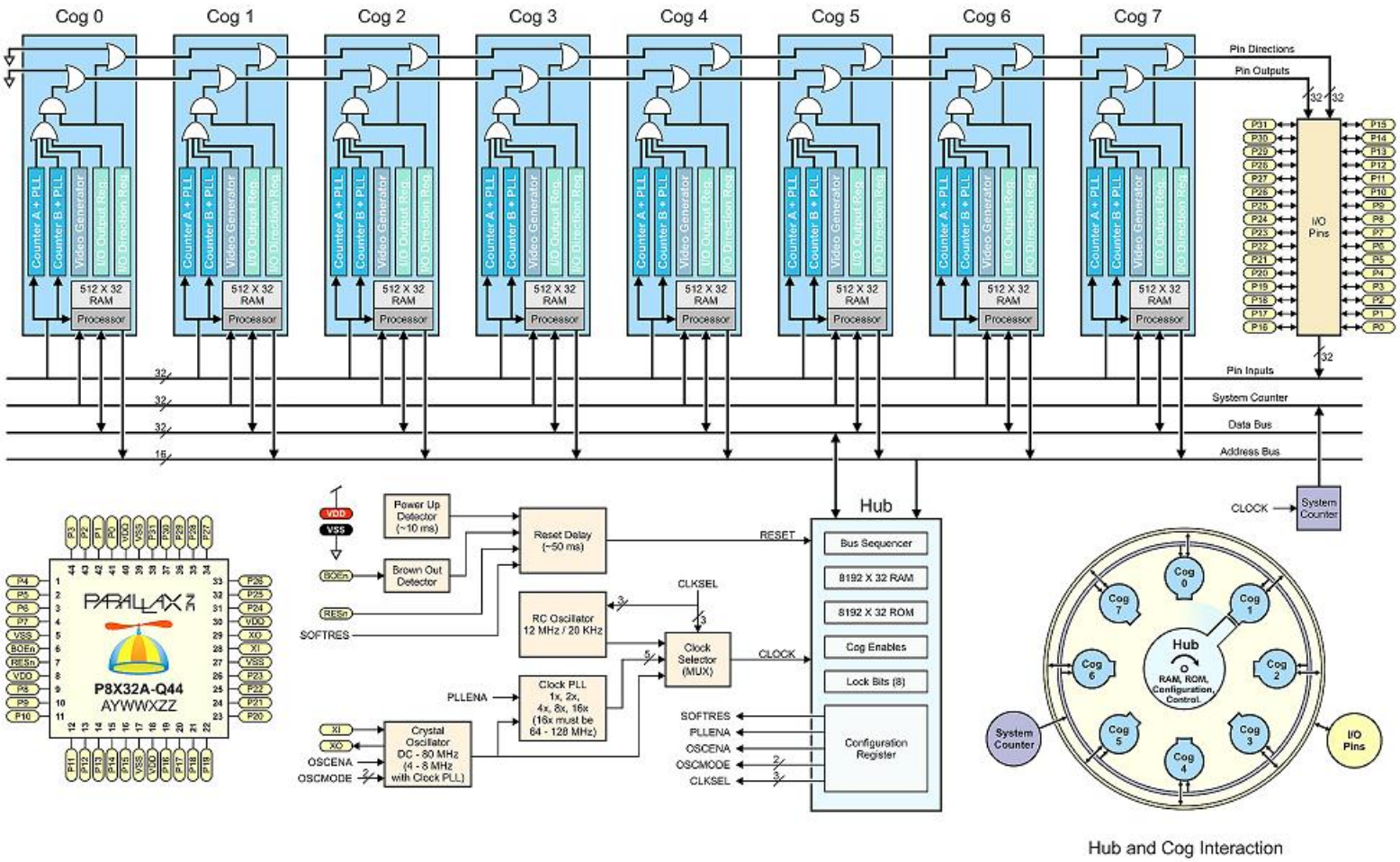
Propeller 1



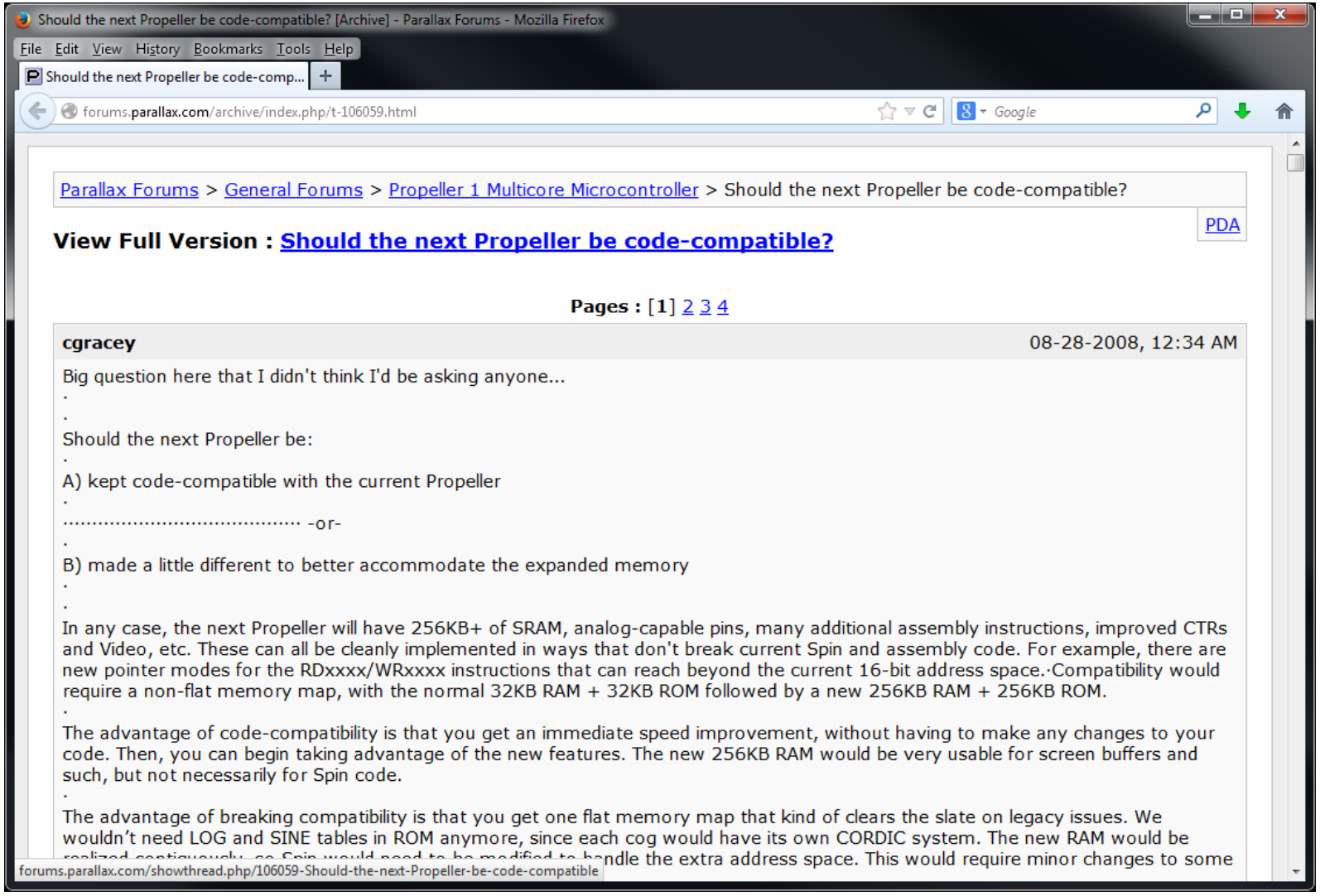
Would you believe this took 8 years to make? Every polygon is ours.

Jeff and I at the Micrion FIB machine. It modifies chips with an ion beam and tungsten gas – like wire cutters, wire, and solder for sub-micrometer work.

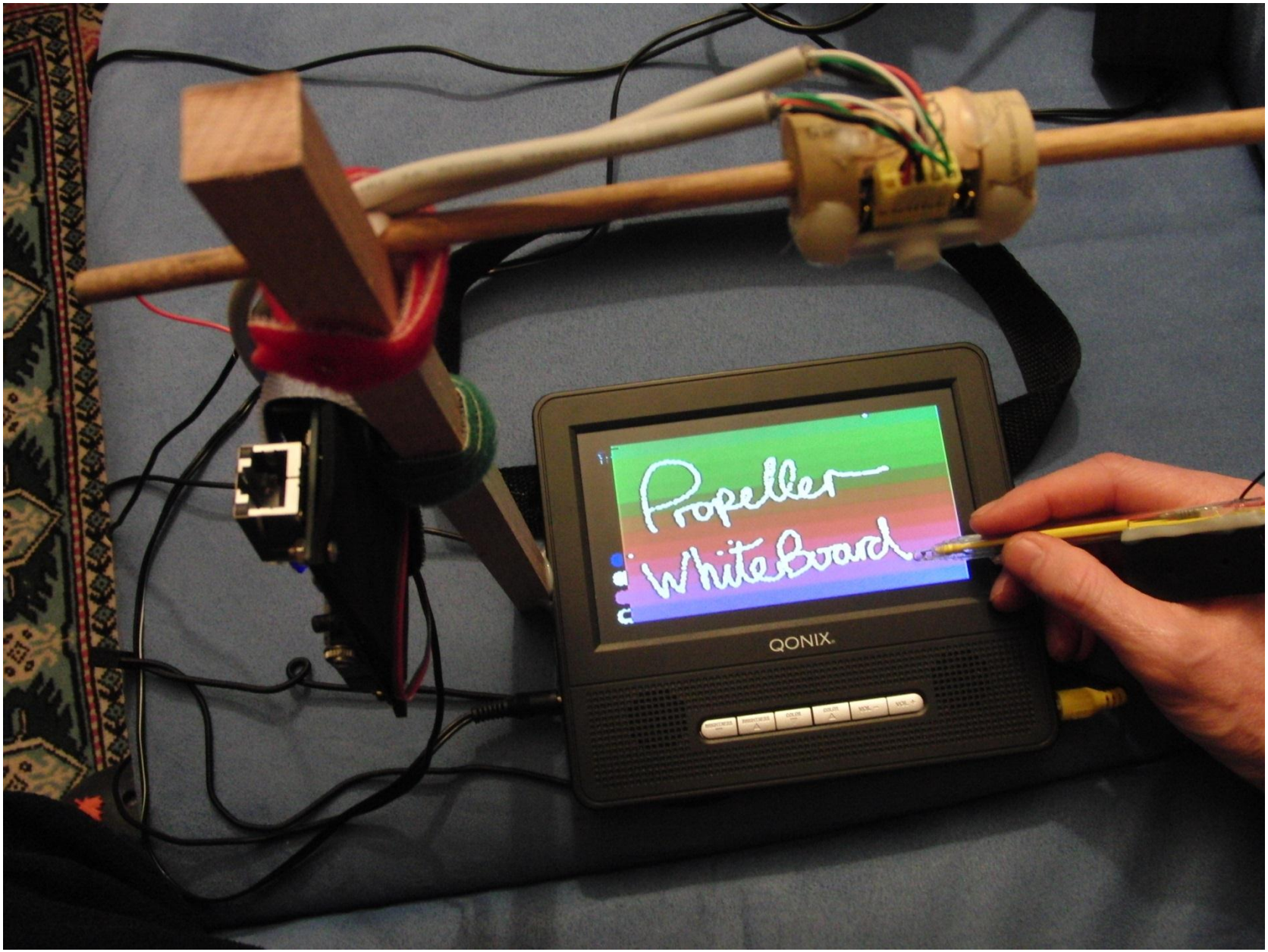
Propeller 1



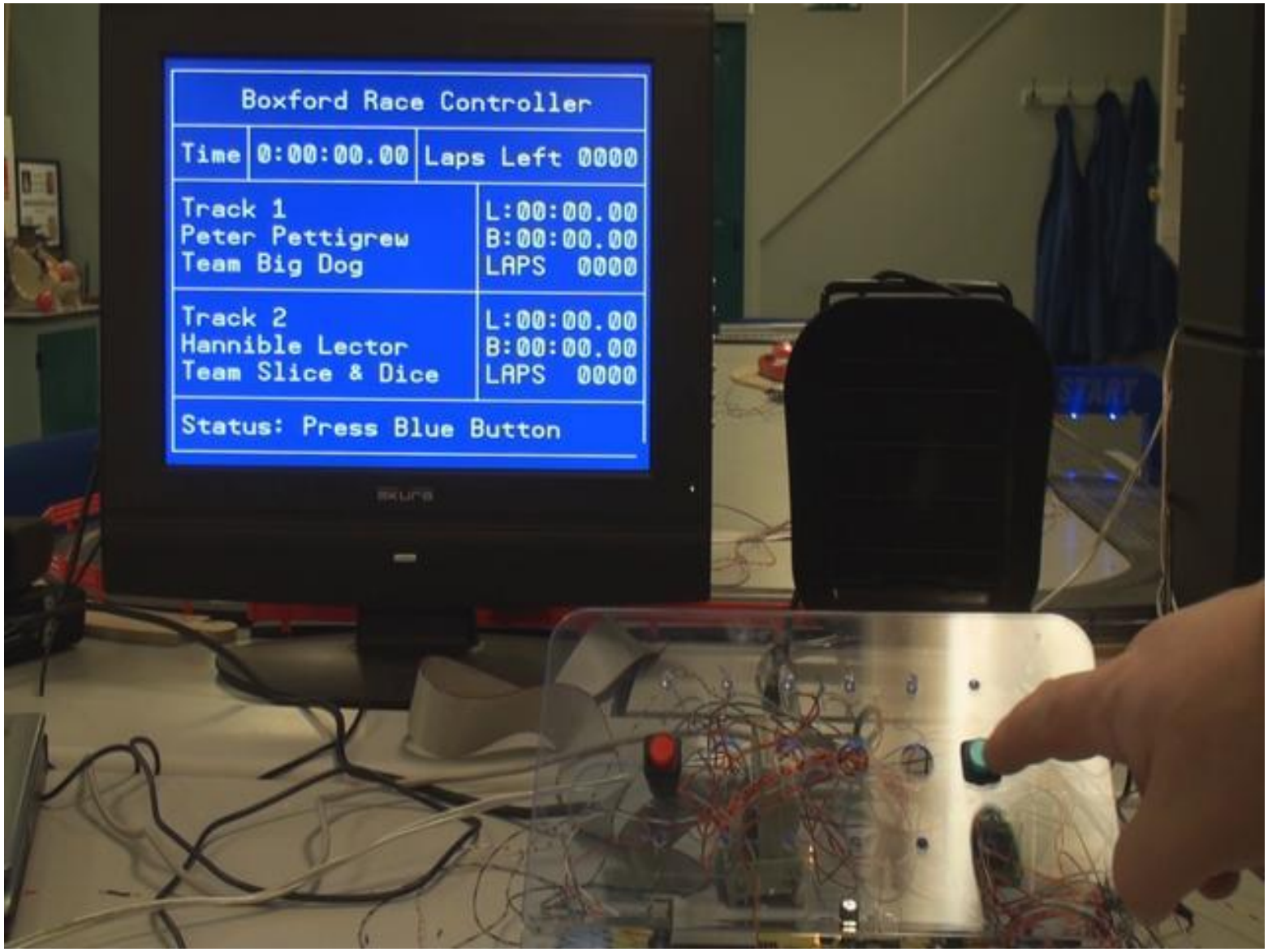
Hub and Cog Interaction



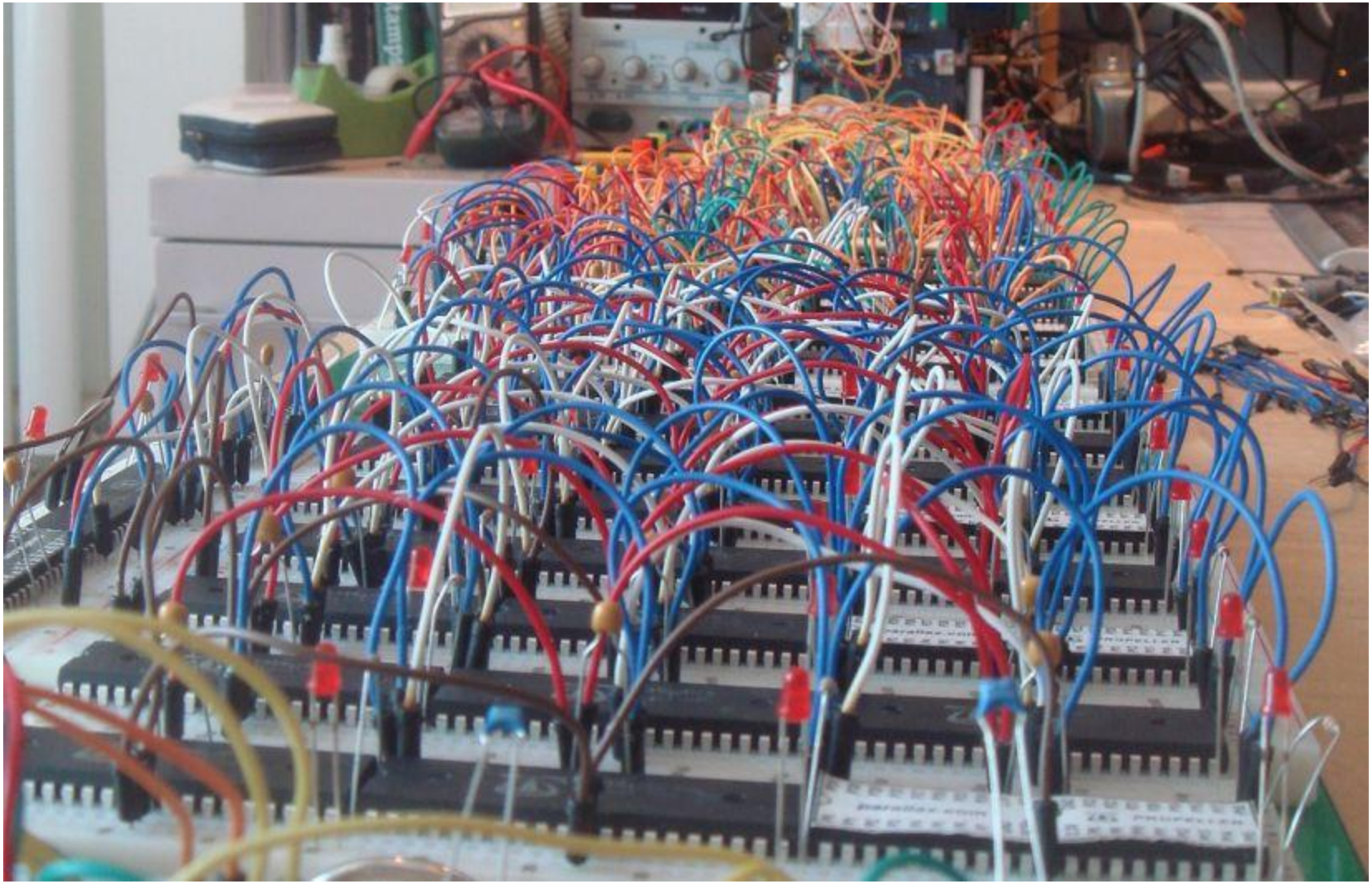
Early forum interaction – I asked questions about what direction to take and the users had lots of great insight on how to improve the Propeller chip.



Someone made this out of a Propeller chip.



Slot car track monitor and scoreboard.



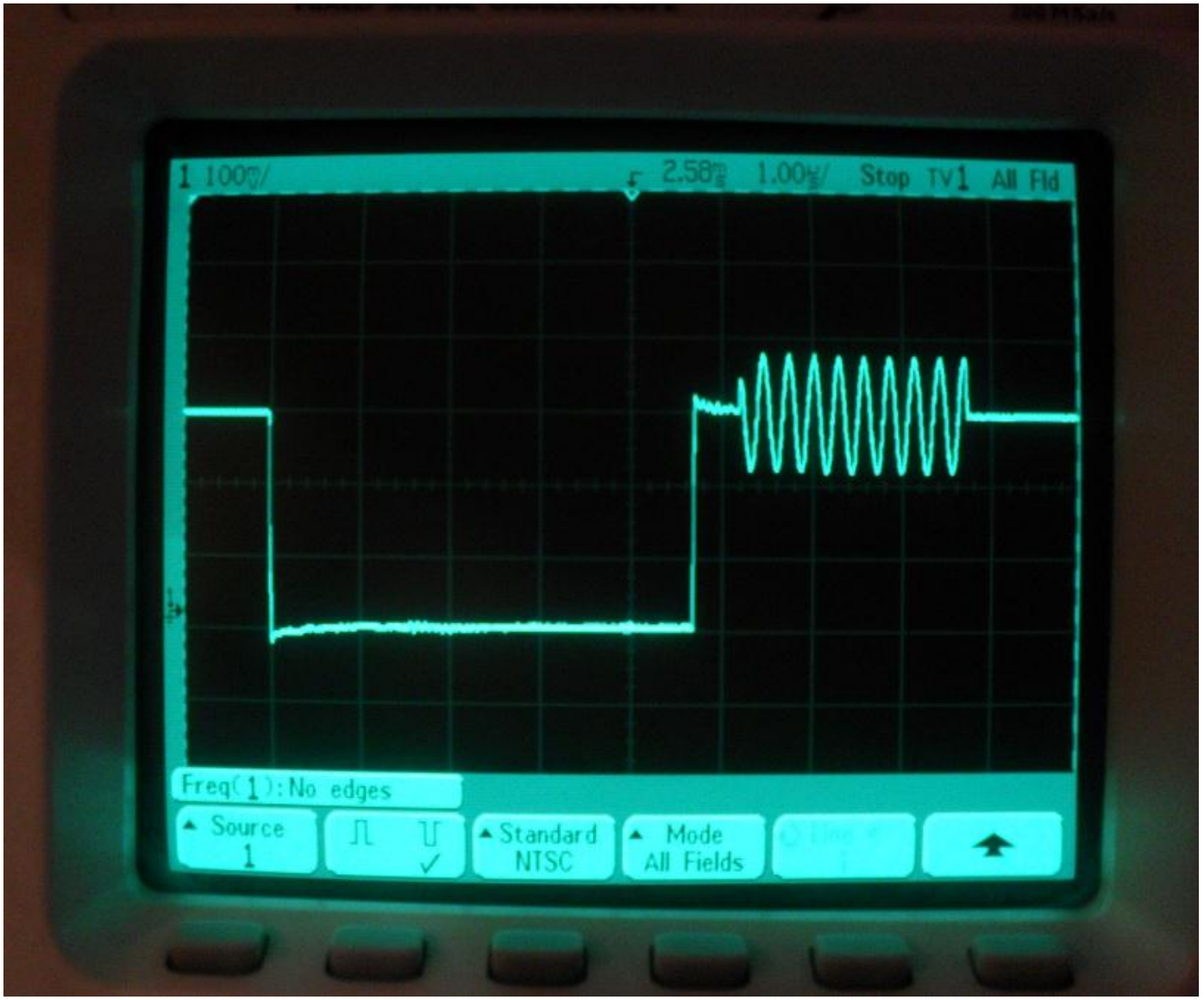
Some are more adventurous than others.

Jeff Ledger from the forum started putting on “Unofficial Propeller Conferences”.

The 1st “Unofficial Propeller Conference” was in Ohio.

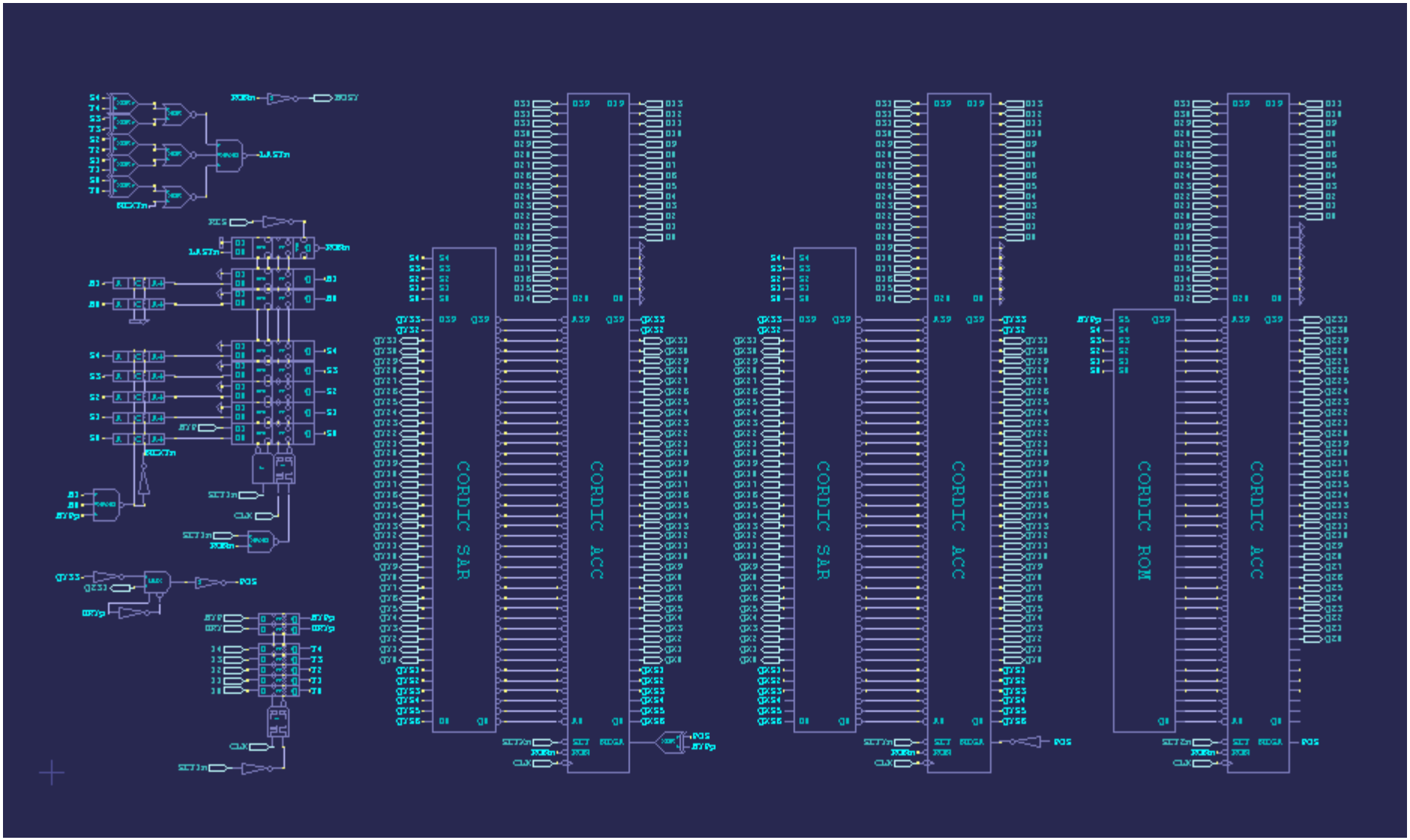
What a delight to discover the Propeller had attracted all kinds of nice people. If you skimmed an embedded-systems trade show for the kindest people, here's who you'd get. How did this happen?

Was this some inadvertent filter?

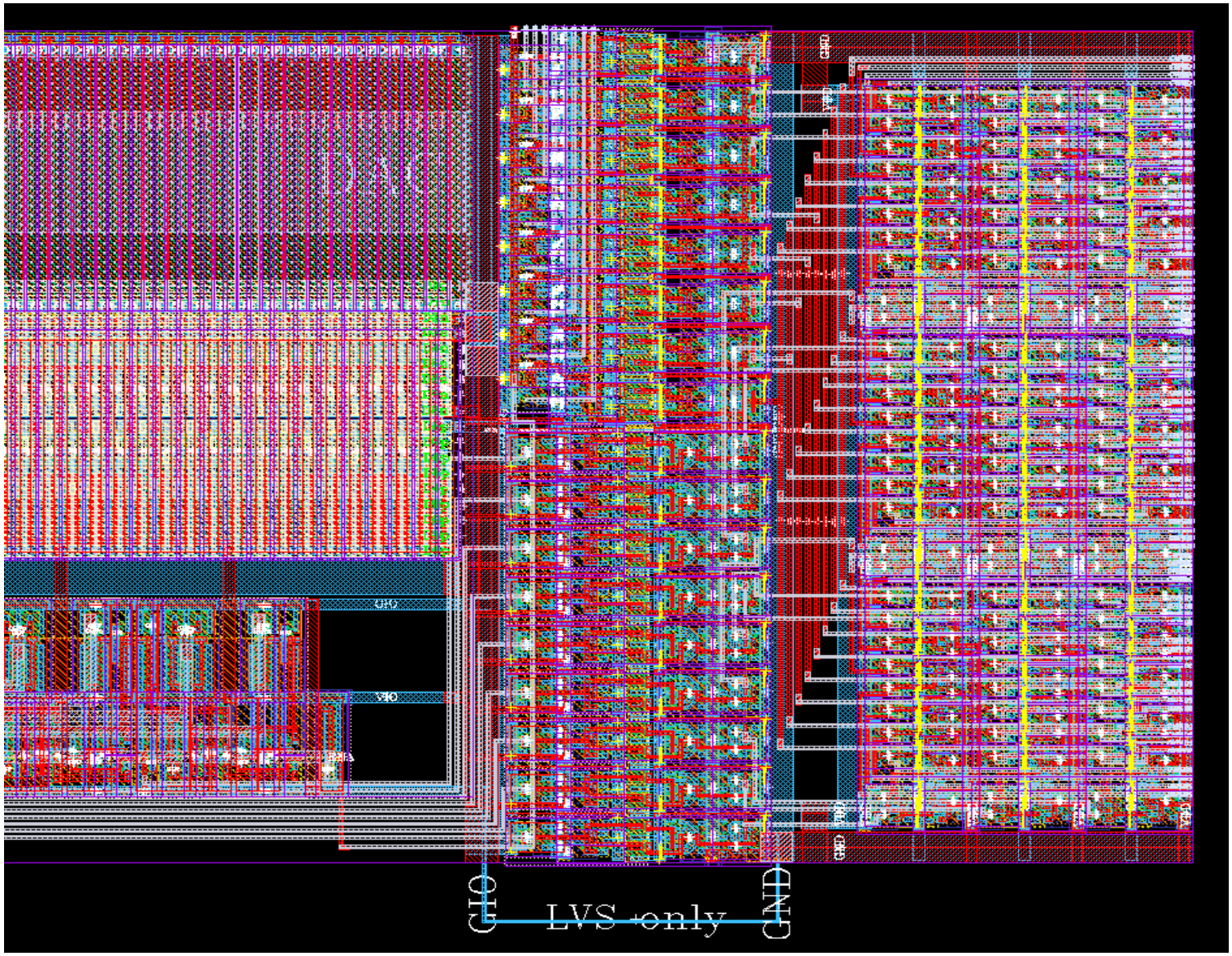


Propeller 2 moves forward with 75-ohm DACs on every pin. Much more analog-friendly.

Now HDTV, thanks to input from an early Propeller Expo attendee.



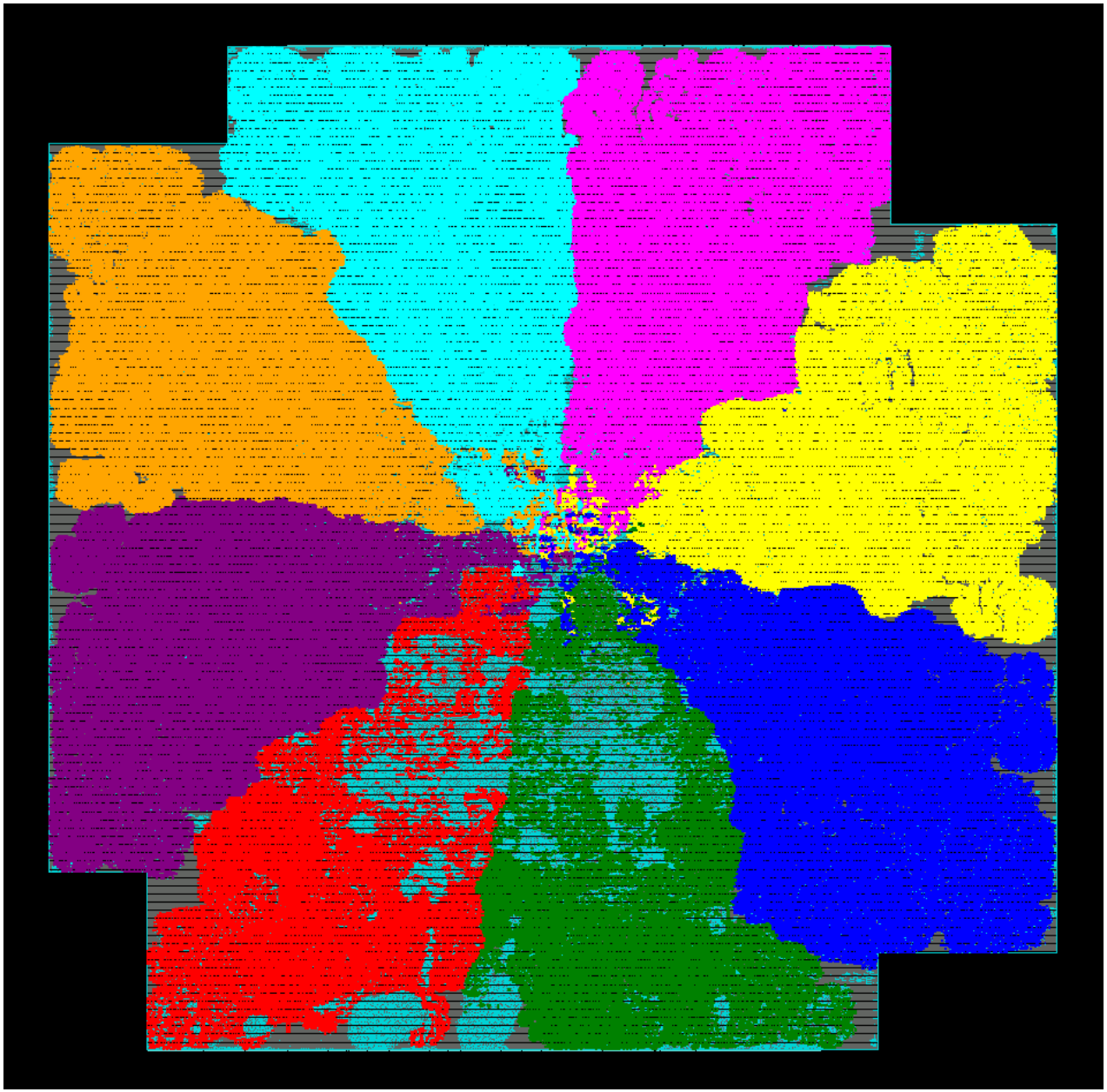
Schematics. Schematics. Schematics for everything.



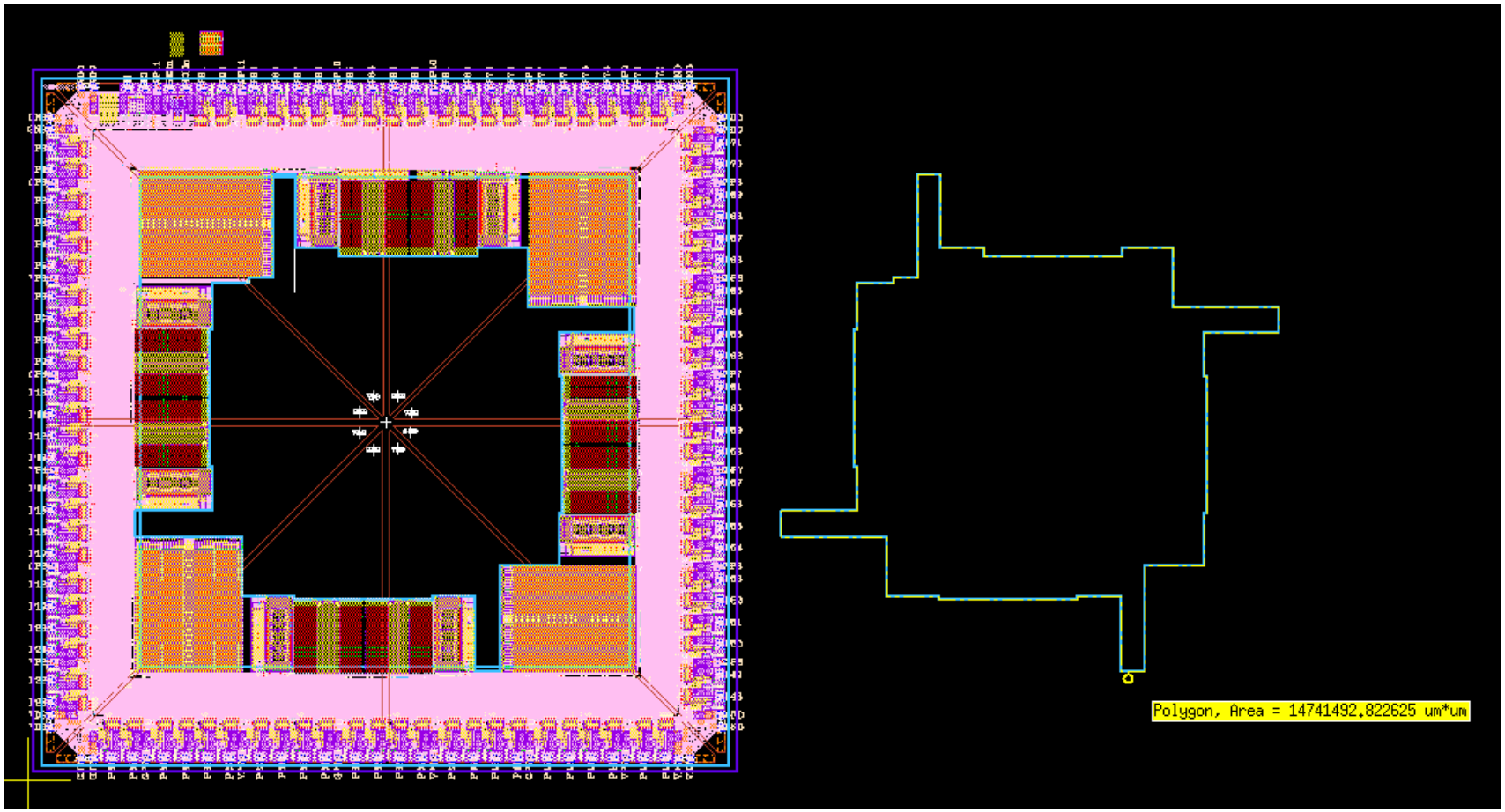
Layout effort is becoming overwhelming. Need to do something different!

```
70
71  wire postadjf       = state == 3'b110;  // fast signal for critical path (eliminates cond qualifier)
72
73  always @(posedge clk or negedge ena)
74  if (!ena)
75      state <= 3'b000;
76  else if (setqz || start || state[2])
77      state <= start      ? {2'b10, qexp_i || qsincos_im_i}   // start         --> iteratex2 or iterate if exp/sincos_im
78             : iteratex2  ? 3'b101                            // iteratex2     --> iterate
79             : iterate    ? last ? 3'b110 : 3'b101            // iterate       --> iterate or postadj if last iteration
80             : postadj    ? lnx && !vec ? 3'b111 : 3'b000     // postadj       --> postadd if exp, else done
81             :              3'b000;                           // postadd/setqz --> done
82
83
84  // prepare log/sincos/arctan/rotate in 1st clock
85
86  wire [29:0] px      = qsincos_i ? s[29:0] : d[29:0];                          // qsincos/qarctan/qrotate
87  wire [29:0] py      = qsincos_i ? 30'b0   : s[29:0];                          // qsincos/qarctan/qrotate
88
89  wire [29:0] pxn     = -px;
90  wire [29:0] pyn     = -py;
91
92  wire [31:0] pmask   = qlog_i ? {1'b0, d[31:1]}                                // qlog
93                      :          {1'b0, (px[29] ? pxn[28:0] : px[28:0]) |       // qsincos/qarctan/qrotate
94                                        (py[29] ? pyn[28:0] : py[28:0]), 2'b11};
95  wire [5:0] pmag;
96
97  mnc mnc_            (.d(pmask), .q(pmag));                                    // encode magnitude
98
99  wire [31:0] zr      = qrotate_i ? z[31:0] : d;                                // qsincos/qrotate
100
101 wire negxy          = !m && (qarctan_i ? d[29] : ^zr[31:30]);                 // qsincos/qarctan/qrotate
102
103 wire [29:0] pxp     = negxy ? pxn : px;                                       // qsincos/qarctan/qrotate
104 wire [29:0] pyp     = negxy ? pyn : py;                                       // note: zr -> z[31:0], pxp -> x[29:0], pyp -> y[29:0]
105
106
107 wire [4:0] limx     = |i                  ? i                                 // iteration override
108                     : qlog_i || qexp_i    ? 5'b11111                          // qlog/qexp
109                     : qsincos_im_i        ? s[4:0]                            // qsincos_im
110                     :                       pmag[4:0];                        // qsincos/qarctan/qrotate
111
112 wire [4:0] sarx     = qexp_i              ? ~d[31:27]                         // qexp
113                     : qsincos_im_i        ? ~s[4:0]                           // qsincos_im
114                     :                       ~pmag[4:0];                       // qlog/qsincos/qarctan/qrotate
115
116 always @(posedge clk)
117 if (start)
118 begin
119     lnx <= qlog_i || qexp_i;
120     hyp <= qlog_i || qexp_i || m;
121     vec <= qlog_i || qarctan_i;
122     arx <= d[29];
```

Verilog and synthesis are the way out of the schematic/layout trap.



8 cogs find their own area within the place-and-route tool.

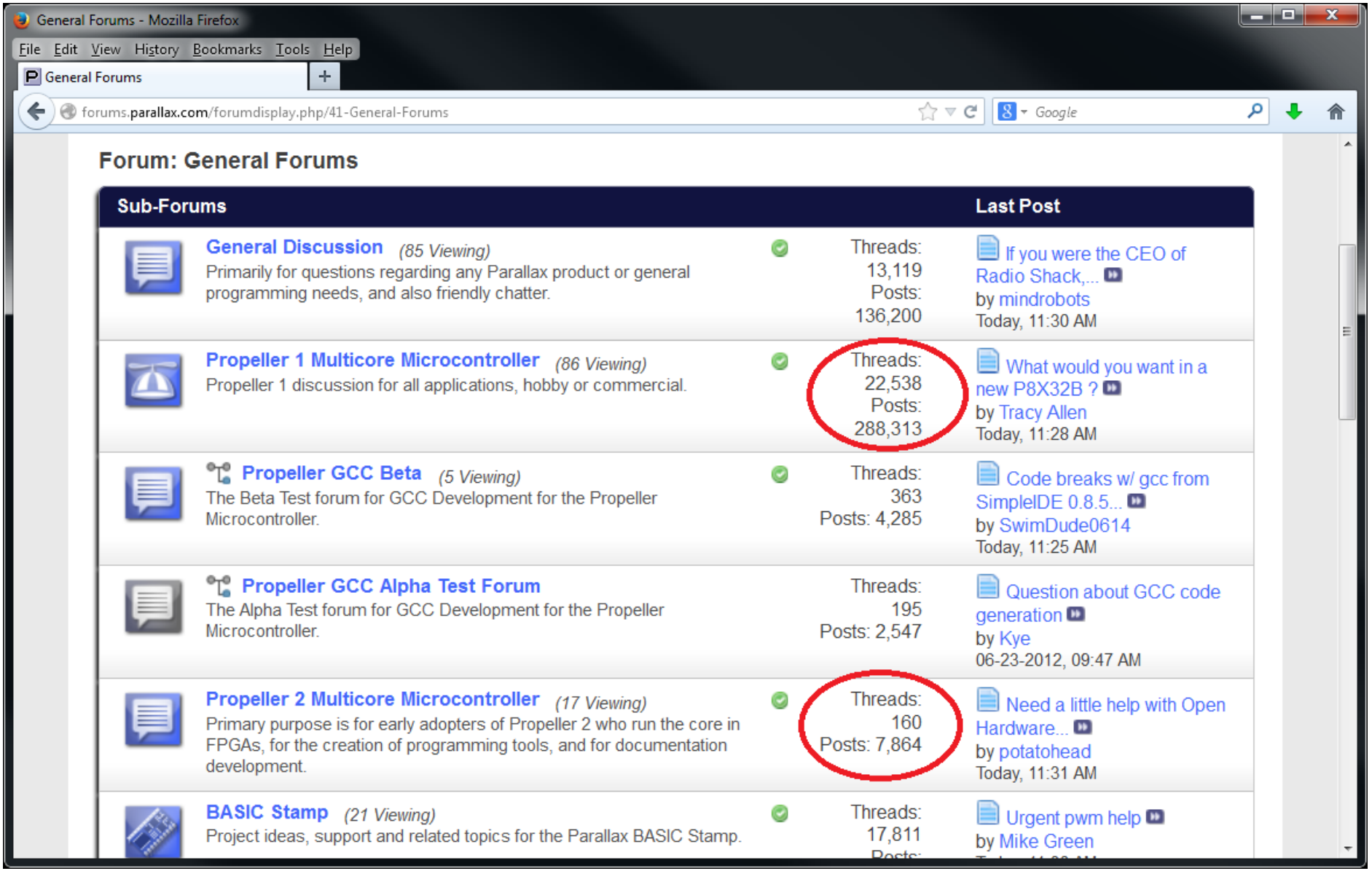


Our full-custom layout with a big hole in the middle for the blob of 500,000 synthesized gates. Nobody could route such a rat's nest on his own. The 3,700 signal connections to our custom layout, plus over 2,000 power connections would be the trivial part.

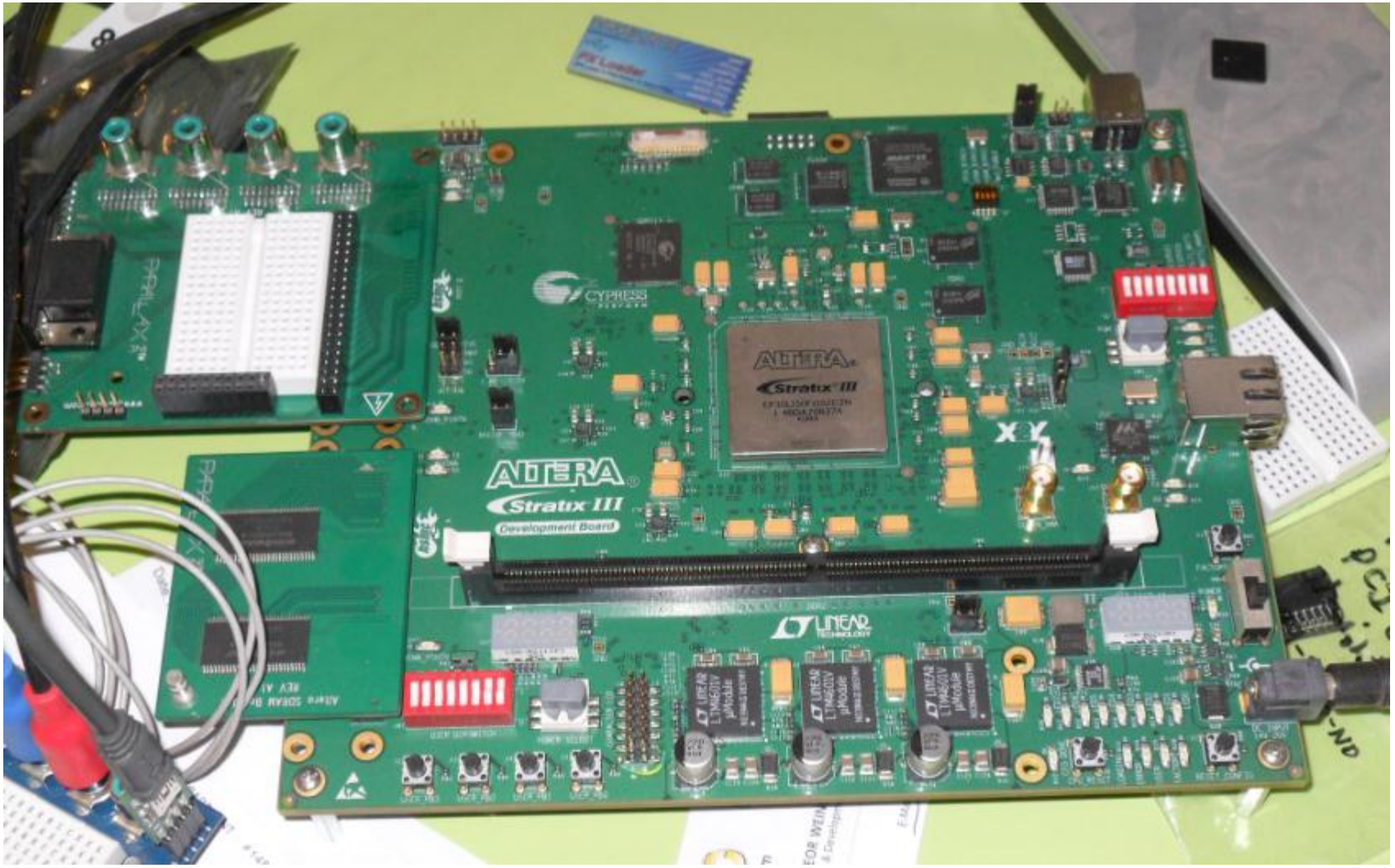
Lots of presenters at a Parallax Expo. Propeller 2 is four years late.
Everyone had a great time.

My dad and Bill Henning at a Propeller Expo.

Gatling paintball gun controlled by a Propeller, made by a special-effects artist.

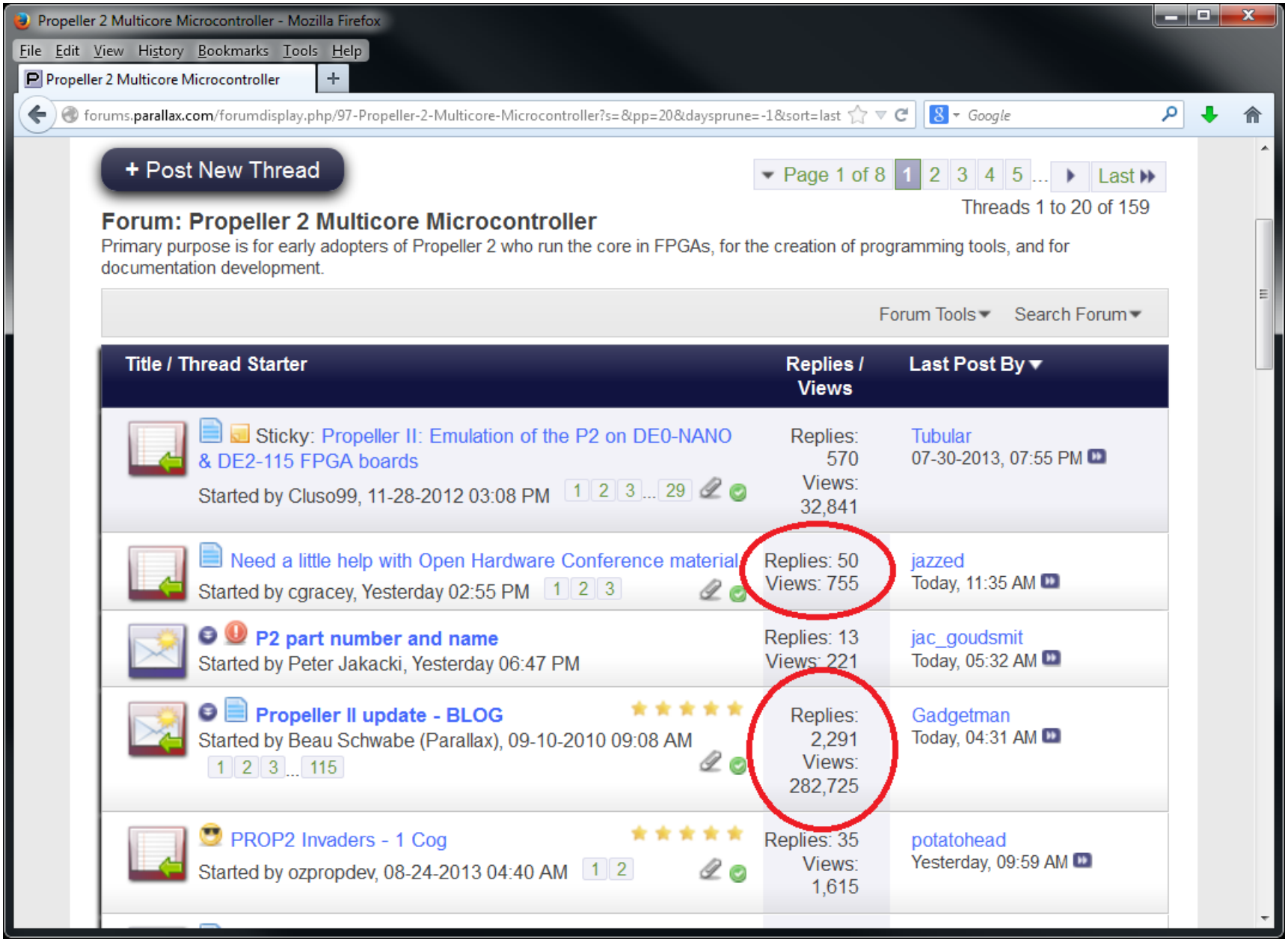


288,000 posts on the main Propeller forum. 7,800 on the early-adopter Propeller 2 forum.



Expensive, but *FAST*, Altera board - $2495 (that Stratix III chip sells for over $3000!?!)
This is what I was using. We had to buy a new one each year to keep Quartus II current.

New Terasic Altera Cyclone IV boards - $595 (5 cogs) or $79 (1 cog).
Everybody can play with Propeller 2 now, before the silicon is done!



2,291 posts into the “Propeller II update – BLOG” with 282,725 views!

SOME PROPELLER 2 CHIP FEATURES RESULTING FROM FORUM INTERACTION

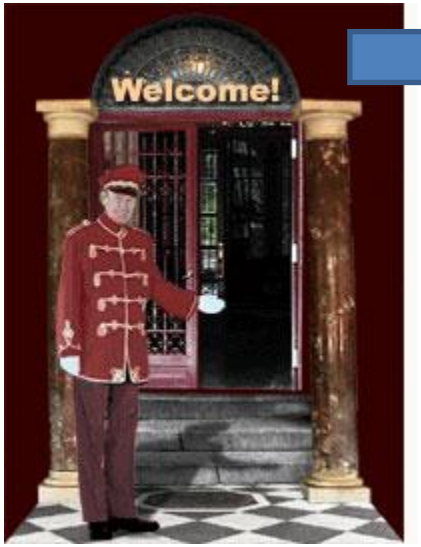
Code protection using SHA-256/HMAC – open standard which is considered secure (Pedward)

Texture mapping for perspective-correct, lighted, blended polygons - 3D graphics!!! (Andre LaMothe and Roy Eltham)

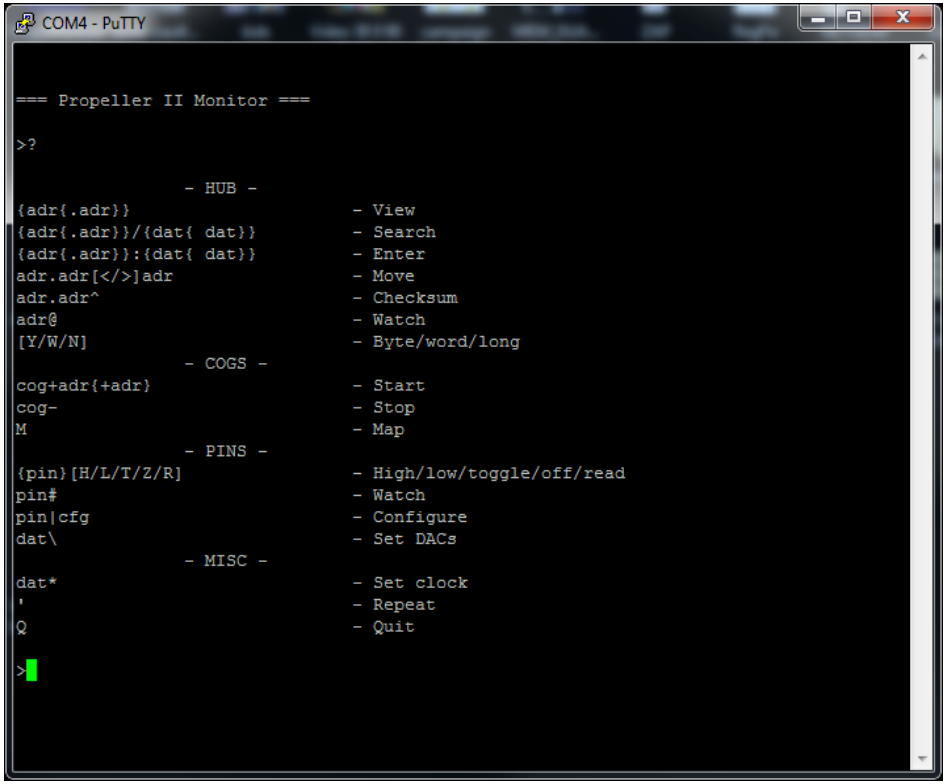
Multithreading – each cog can now run 4 different programs simultaneously. (Heater, Bill Henning, Ariba, and others)

Friendly boot monitor - no need for a development system if you just want to load some code or poke around.
(Potatohead, others)

The dreaded code-protect issue finally gets solved, elegantly!



Security bits are clear.
Welcome, Everyone!



You'll have to sign your loader if you want to mess with those bits, Sir.

Bits are set!
You'll get nowhere without a proper signature.

12-10-2012, 12:26 PM #32

Ariba

Posts: 1,530

Re: LMM2 - Propeller 2 LMM experiments

I've tried a few hours, with all variations, but can't get the QUAD-LMM code to work. Here is the simplest version:

Code:

```
' Quad LMM
DAT
        org 0
        mov pc,start
lmm     reps #511,#6
        setquad #ins1
         rdquad pc
ins1     nop            'quad aligned
         nop
         nop
         nop
         add pc,#16
        jmp #lmm        'after 511 repeats
```

12-10-2012, 09:39 PM #80

cgracey

STAFF

Location: California
Posts: 2,001

Re: LMM2 - Propeller 2 LMM experiments

I see the problem in the Verilog code now. I'm using an address to compare that is not aligned with the proper pipeline stage.

I'm going to fix this and post new files for the Terasic boards. It's going to take three days of synthesis/layout/verification to fix this. Better now than later.

Thanks for discovering this, Bill, Ariba, and any others who've worked on this!

Reply With Quote

Nasty bug gets discovered and fixed right before a tape-out. Notice that 48 posts occurred in 9 hours between discovering and identifying the bug. Whew!

08-11-2012, 09:35 AM #280

cgracey

STAFF

Location: California
Posts: 2,001

Re: Propeller II

> Originally Posted by **Heater.**
>
> *Chip,*
> *I cannot see the complete code on my phone here but that tasksw looks really sweet. Now that you have a context switching mechanism is there a simple way to get task switch to happen automatically on every instruction? So two tasks would be able to run at half normal rate each. No overhead of having to read and execute a tasksw instruction. To keep it simple there would be no priority mechanism.*
> *In fact it would be nice for the task switch to happen after every instruction time even if the instruction has not finished. Then multiple tasks could be waiting on different events, pin or time or vid.*

I wish I had thought about this earlier, because it might have been somewhat trivial to have an array of 8 program counters and z/c flags that could be switched among. Man, that's pretty compelling! Ask yourself this: if instructions floated through the pipeline that each represented a different pc/z/c, would it matter, as long as appropriate pc/z/c's were updated at the end of each instruction? Would the registers care? I don't think so, but it would take a little consideration to know for sure.

As TASKSW works right now, it's actually a JMPRET instruction, so it takes 4 clocks (1 to execute plus 3 to reload the pipeline).

Reply With Quote

08-28-2012, 10:32 PM #1244

cgracey

STAFF

Location: California
Posts: 2,001

Re: Propeller II

The time slicing is now working in the Prop II. Here is some code and a screenshot of it running at 200MHz. Note that when all four tasks are running, jumps only take one clock, as there's no same-thread instructions in the pipeline to cancel. In this example, each task loops every 8 clocks. Were any of these tasks to run solely, they would take 5 clocks (1 for the NOTP and 4 for the JMP). This way, they take just two clocks each.

Code:

```
PUB go

  coginit(0,@tasks,0)

DAT
```

Thanks to Heater, each cog in the Propeller 2 is now multi-tasking. I was so excited when he thought of this. I worked like mad and 17 days and 964 posts later, it was implemented.

07-02-2013, 04:40 AM #2002

cgracey

STAFF

Location: California
Posts: 2,001

Re: Propeller II update - BLOG

> Originally Posted by **Ariba**
>
> *With an XOR instruction it should be possible to change a field of bits to any state you wish with an atomic instruction:.*
> *Example:*
>
> *Code:*
>
> ```
> outa[9..3] := $55
>
>   mov bits,#$55
>   mov mask,#%1111111
>   shl bits,#3
>   shl mask,#3
>   mov tmp,outa
>   xor tmp,bits
>   and tmp,mask
>   xor outa,tmp    'atomic modify
> ```
>
> *As long as the concurrent PASM code does not change bits 3..9 of outa this should work.*
>
> *Andy*

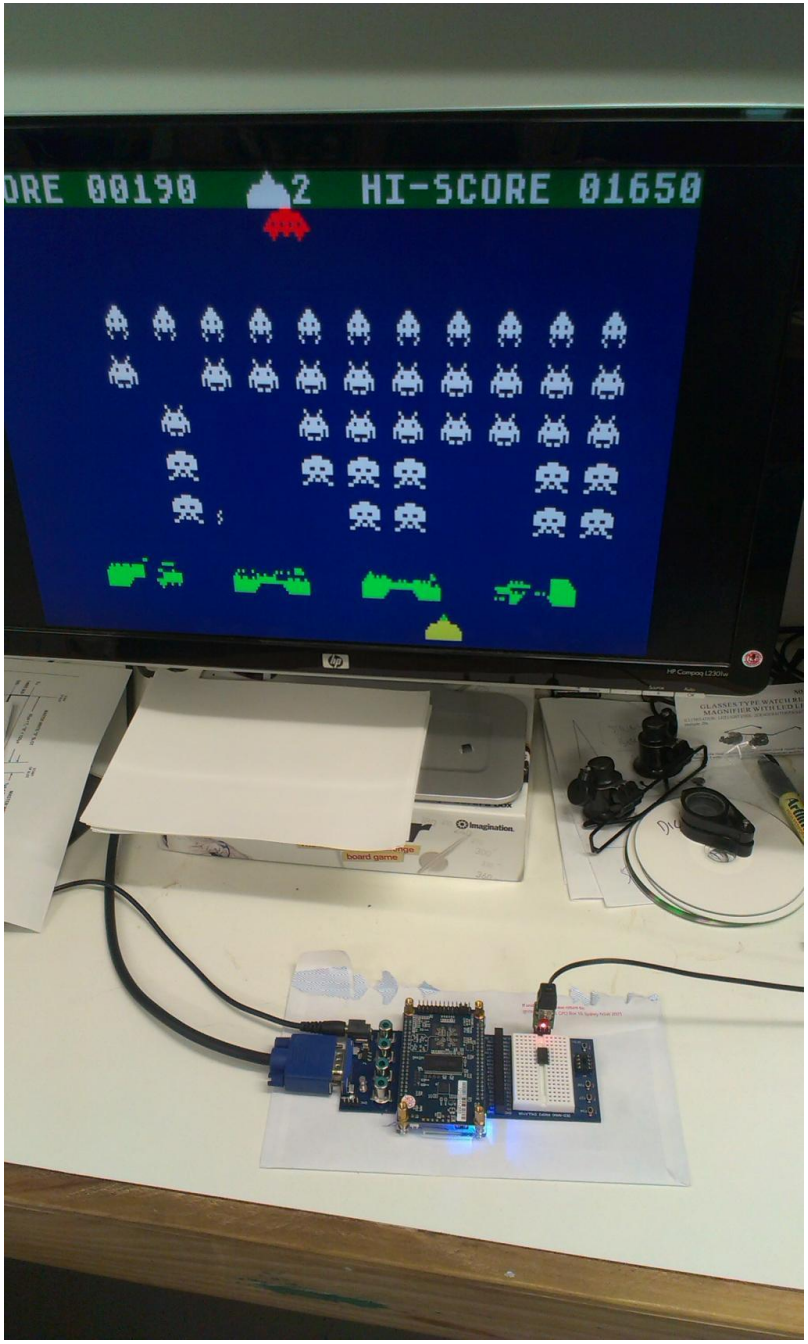
Wow!!! That's a great idea. As long as the Spin code and concurrent PASM code don't mess with each other's pins, there would be no conflict. Problem solved. And no need for new instructions!

Thanks for thinking about this, Andy.

I'll have to make shadow registers for not just PINx, but DIRx, as well.

Reply With Quote

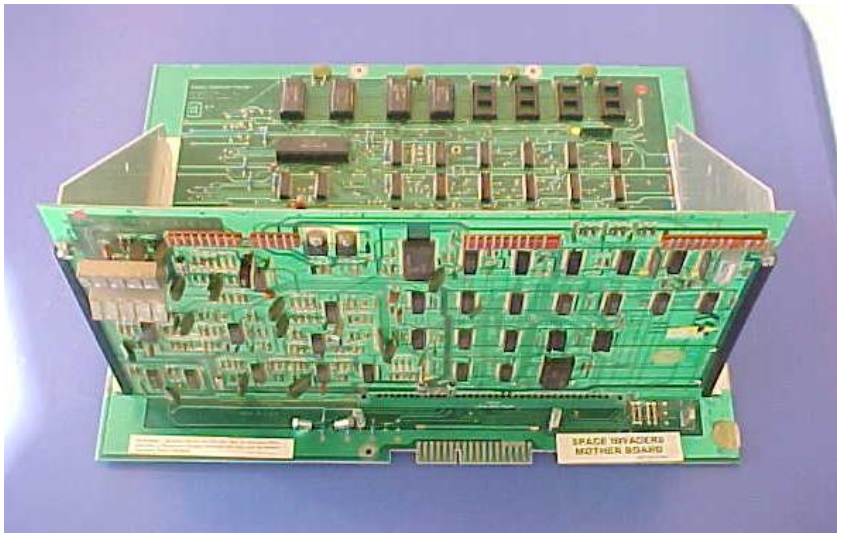
Atomic XOR makes Spin/PASM multitasking compatible. This saved the day.



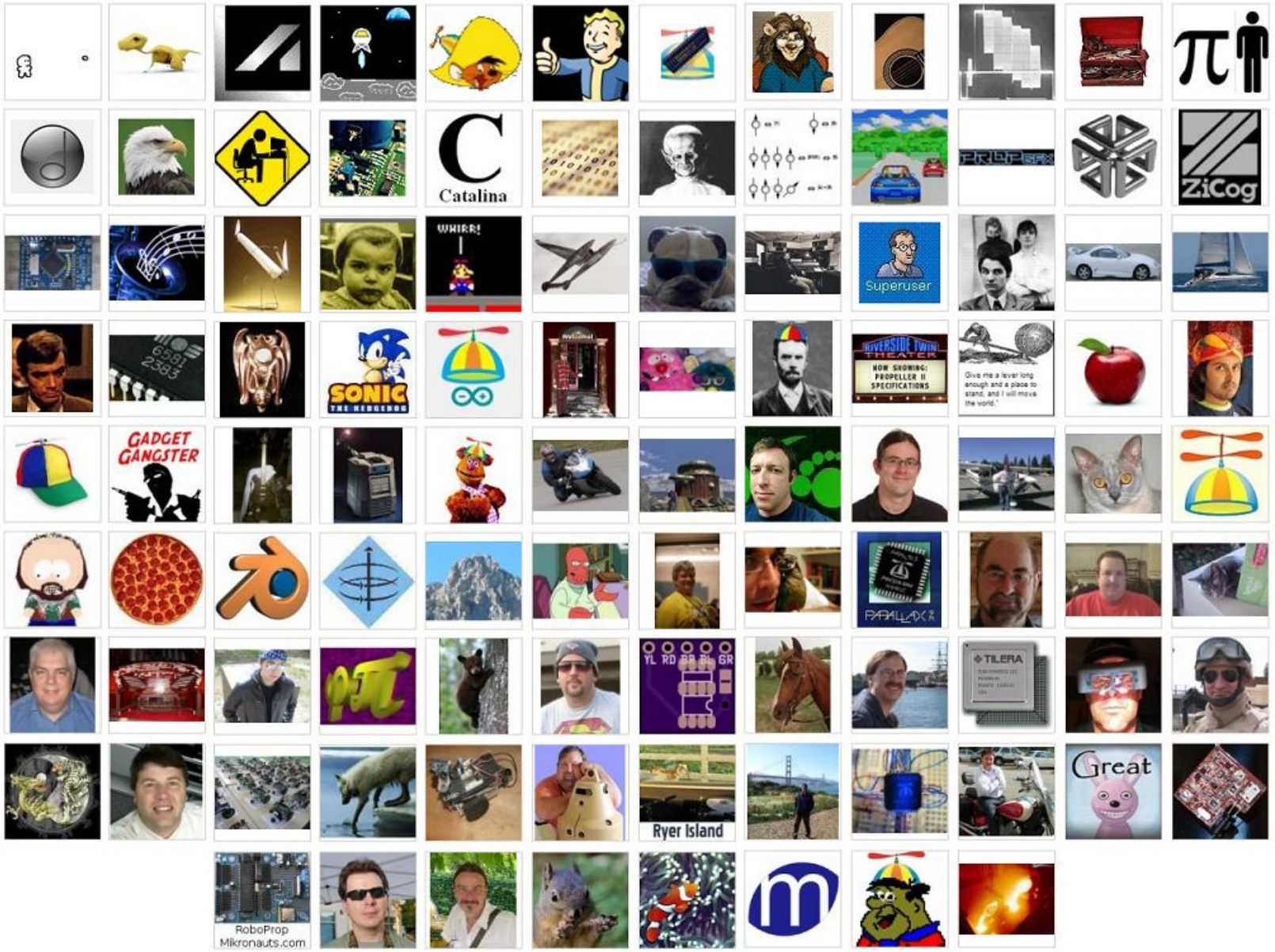
Ozpropdev made a Space Invaders game on the 1-cog DE0-Nano board.

This is 1/8th of a Propeller 2 running at 3/8ths speed. It even has a background serial monitor.

Too bad we can't go back to 1978 with a bunch of Propeller chips.



Here is an original Space Invaders motherboard setup.

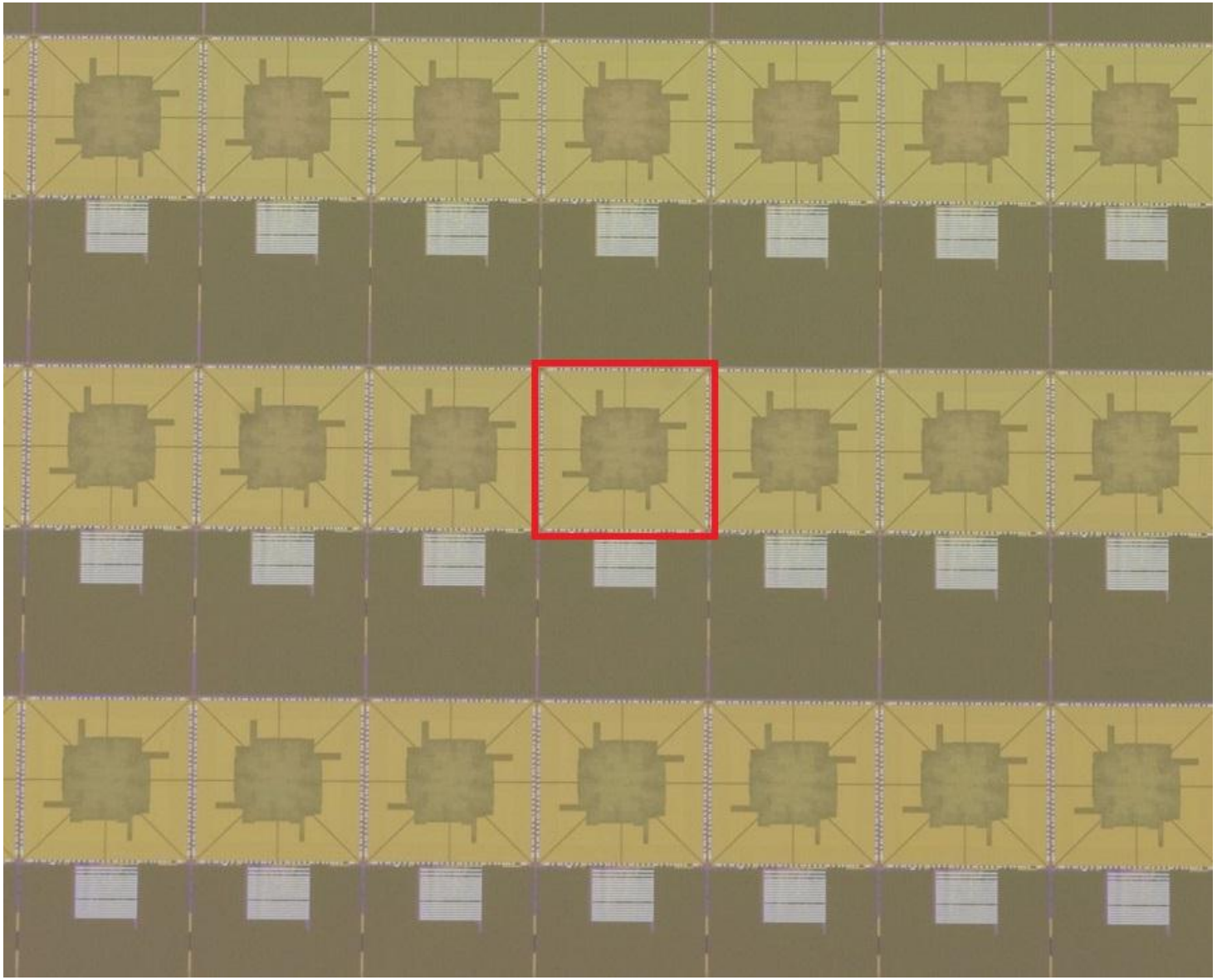


Here's a collage of most of the avatars of the forum members who've been formative in the Propeller 2's development. They are from all over the world.

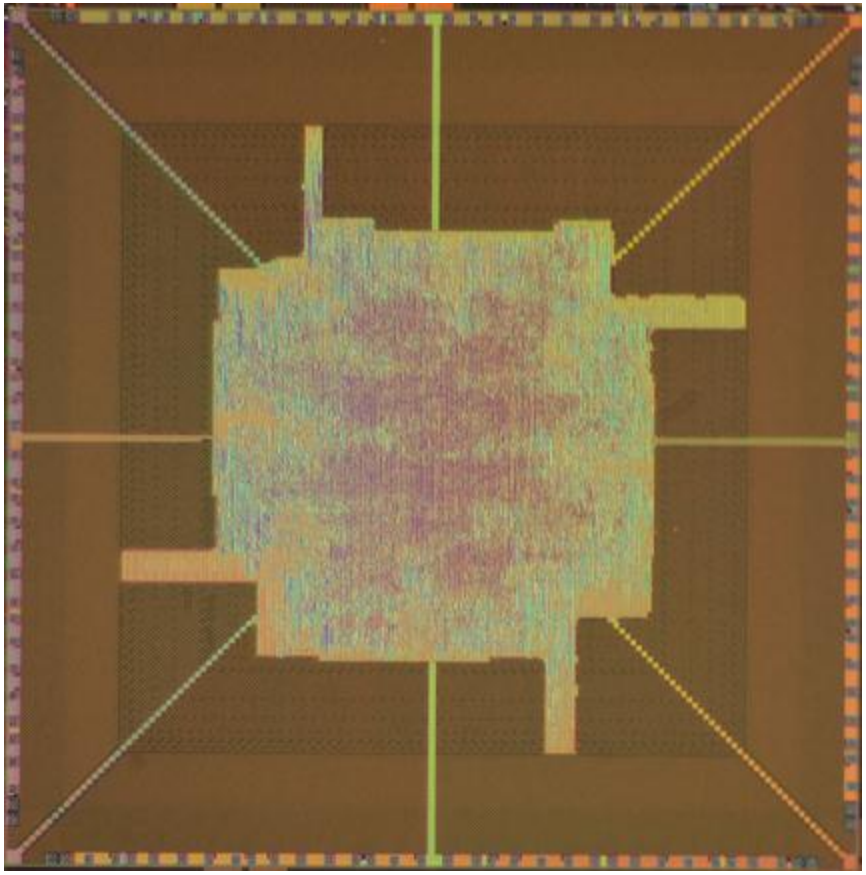
It’s said that a camel is a horse designed by committee.

How to wind up with a horse, after all:

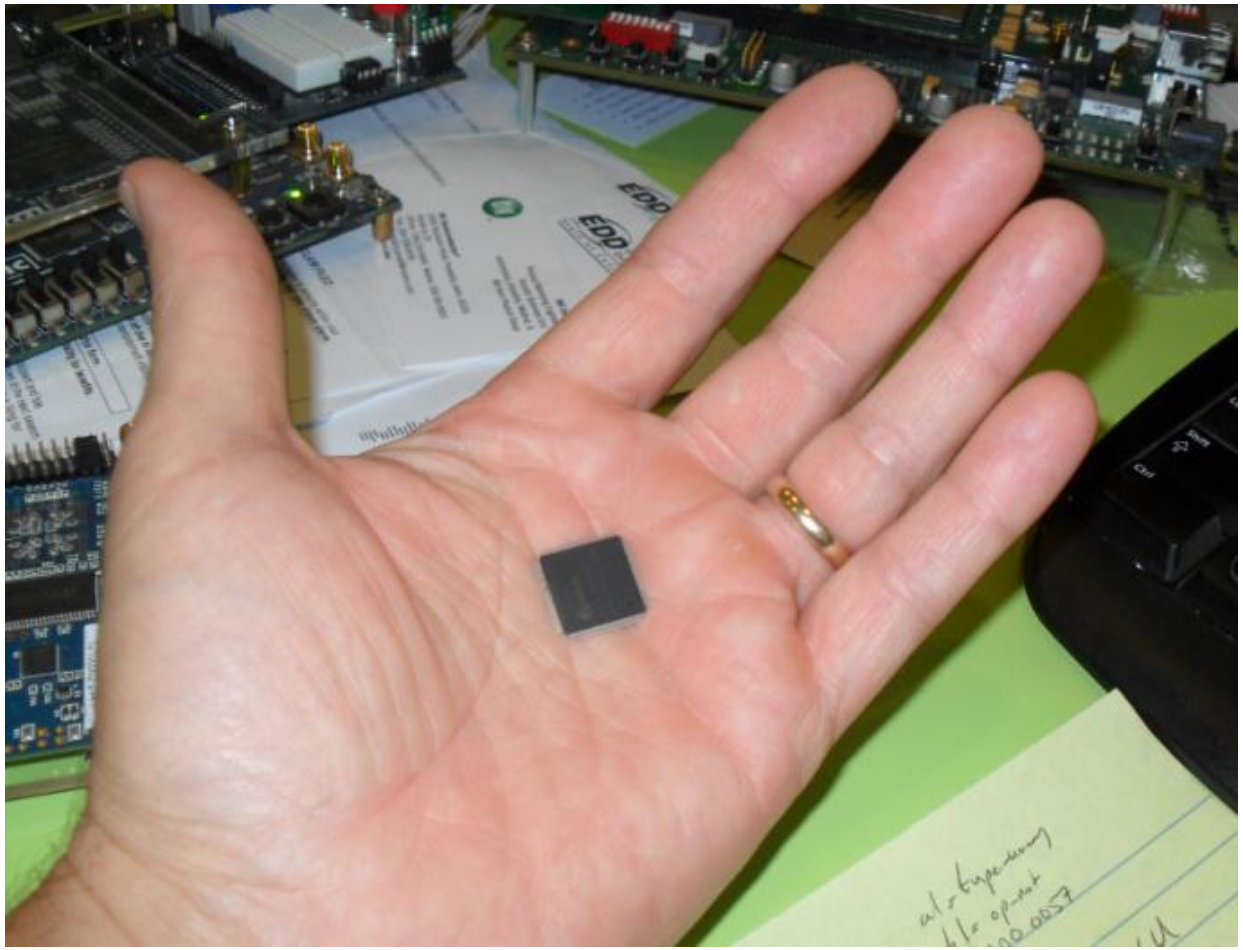
- You need a grounded, gut sensitivity about what you are doing. It develops over *much* time, if you really care.
- You need autonomy to operate. You have to forge that yourself, which usually means working alone, or expecting to.
- Work within a sphere in which you *can* have control, even if it’s quite limited.
- Others of like mind will eventually join you and you will be able to happily collaborate, without dread of compromise, as you are both free agents. Any less freedom could only produce a donkey. Look at that horse!



The new Propeller 2 wafers are back, getting packaged. We hope they’re good.



New Propeller 2 die under a microscope.



If the new wafers are good, we'll have lots of these, at last.
Seems a little anticlimactic, doesn't it?

This is pretty much what a Propeller chip is all about – it's a handy workshop in which you can build amazing things. I'm really looking forward to getting in there!

THE END

Thank you!