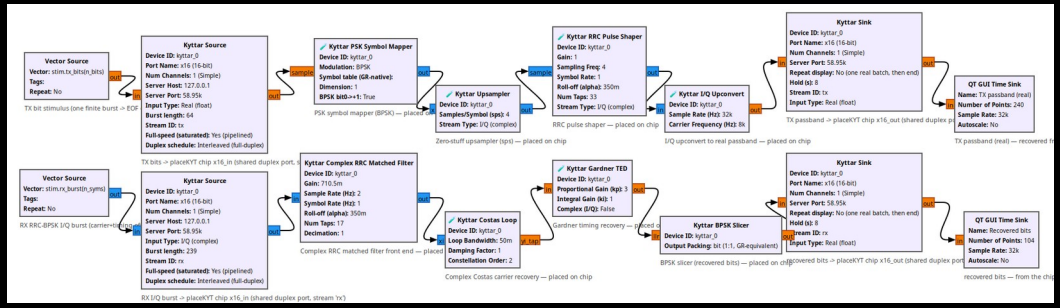
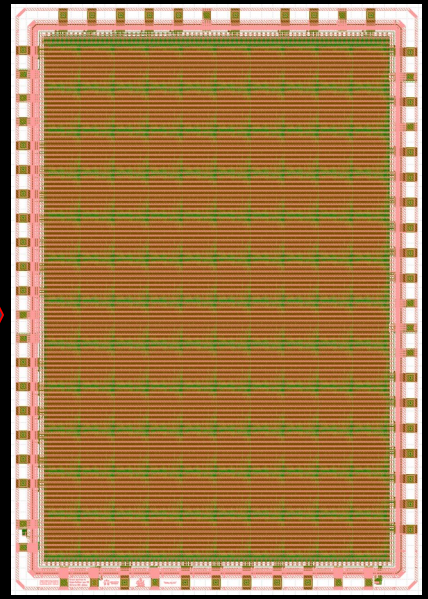




Your Flowgraph is the Chip: GNU Radio Compiled Straight to Silicon

Chuck McClish
CEO and Founder
Lattrex Inc.

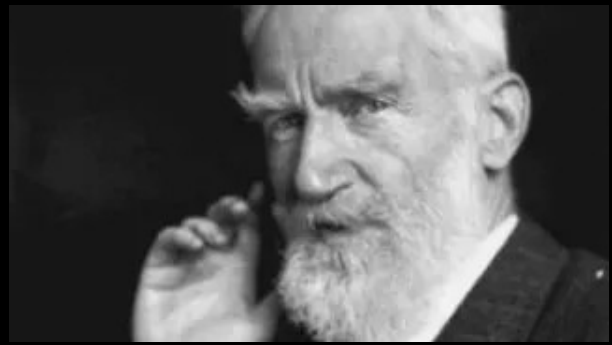
The Kyttar chip

- Began as a thought experiment:
  Is it possible to leverage biological architectures to solve the complexity/scaling problems of today's chip designs?
- Properties of fungal networks:
  - Array of independent, identical cells
  - Each cell is self-contained, there is no global control structure
  - Each cell communicates with its direct neighbors and passes information through to communicate with cells further away
- Kyttar is a computational substrate where programs to live

Yet another MPPA

- I had independently reinvented the MPPA (massively parallel processor array)
  - Tried many times over the last 40 years very few survived

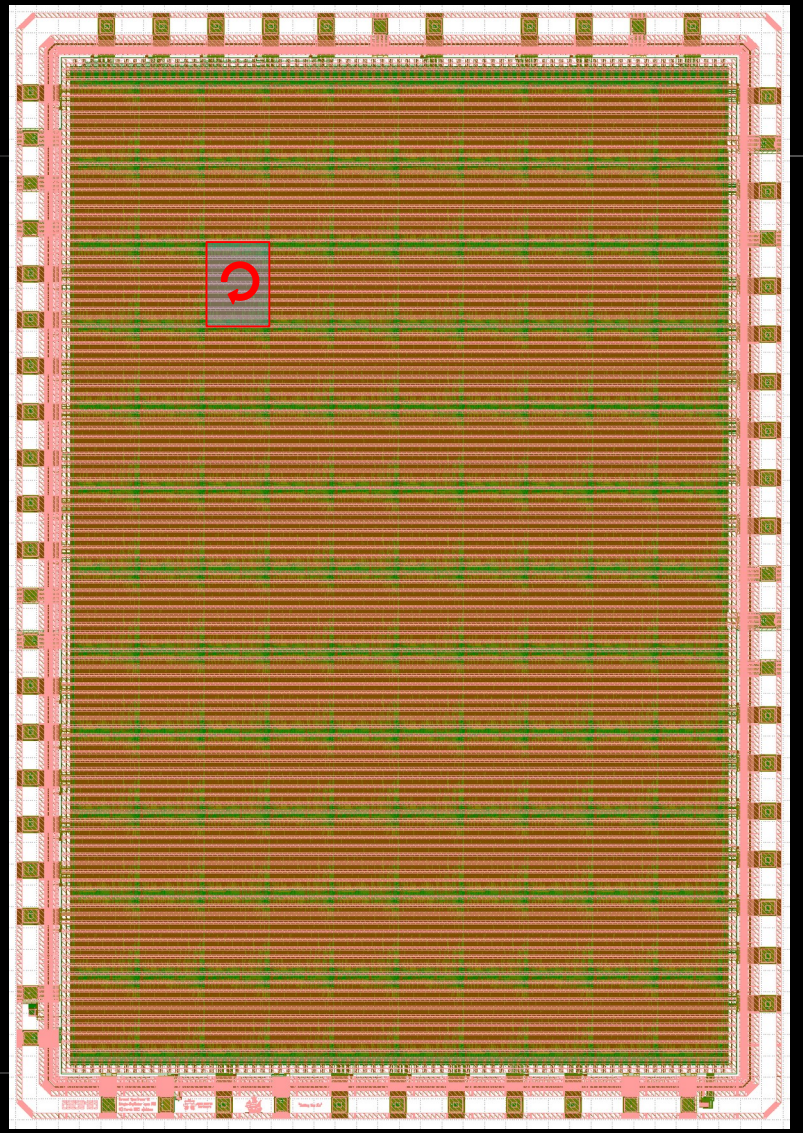
"The reasonable man adapts himself to the world, the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man." – George Bernard Shaw



- I had 3 questions I needed to answer:
  - What fine grained parallel workloads map well to this spatial computation paradigm?
  - How will it be programmed?
  - Who would champion a new architecture like this?

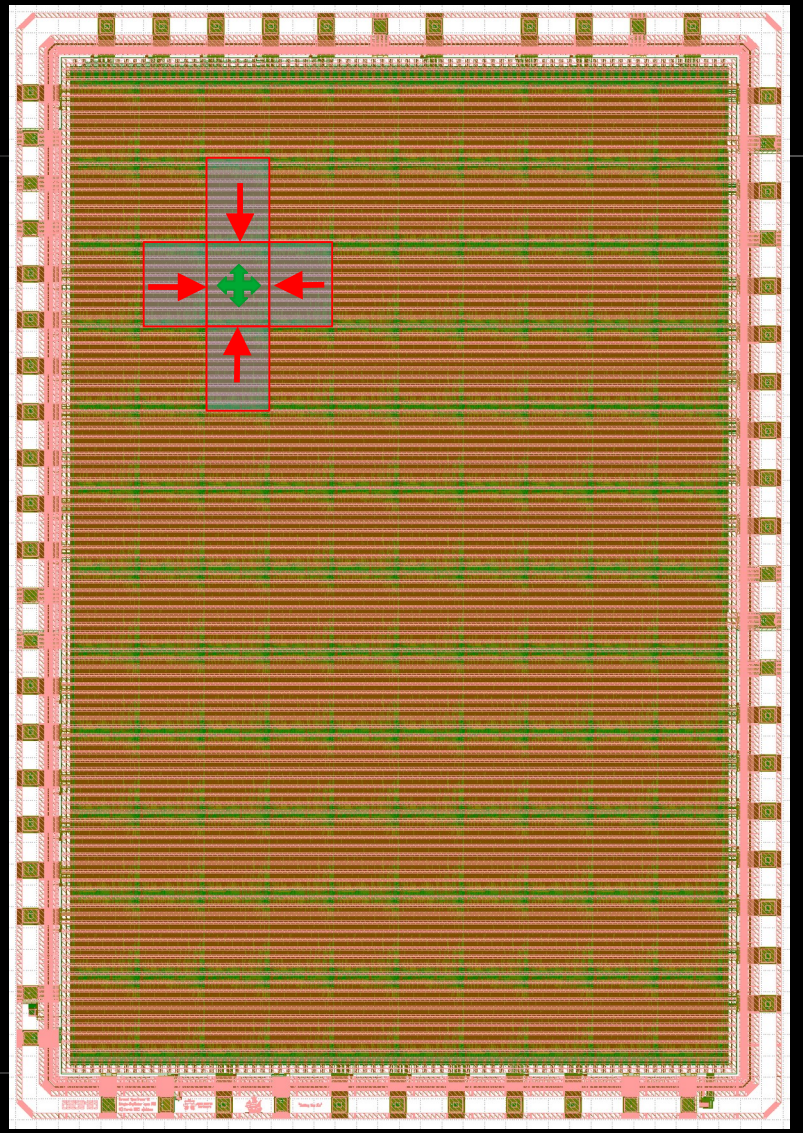
The cell

- Each cell is a small async MISC machine
  - 16 bit instruction and data, 13 instructions
  - 32 words of memory with 0 address as accumulator
  - 16x16 MAC unit, adder, logic functions
  - Executes instructions in its internal memory
- When a cell is executing, it blocks passing data and receiving new incoming data



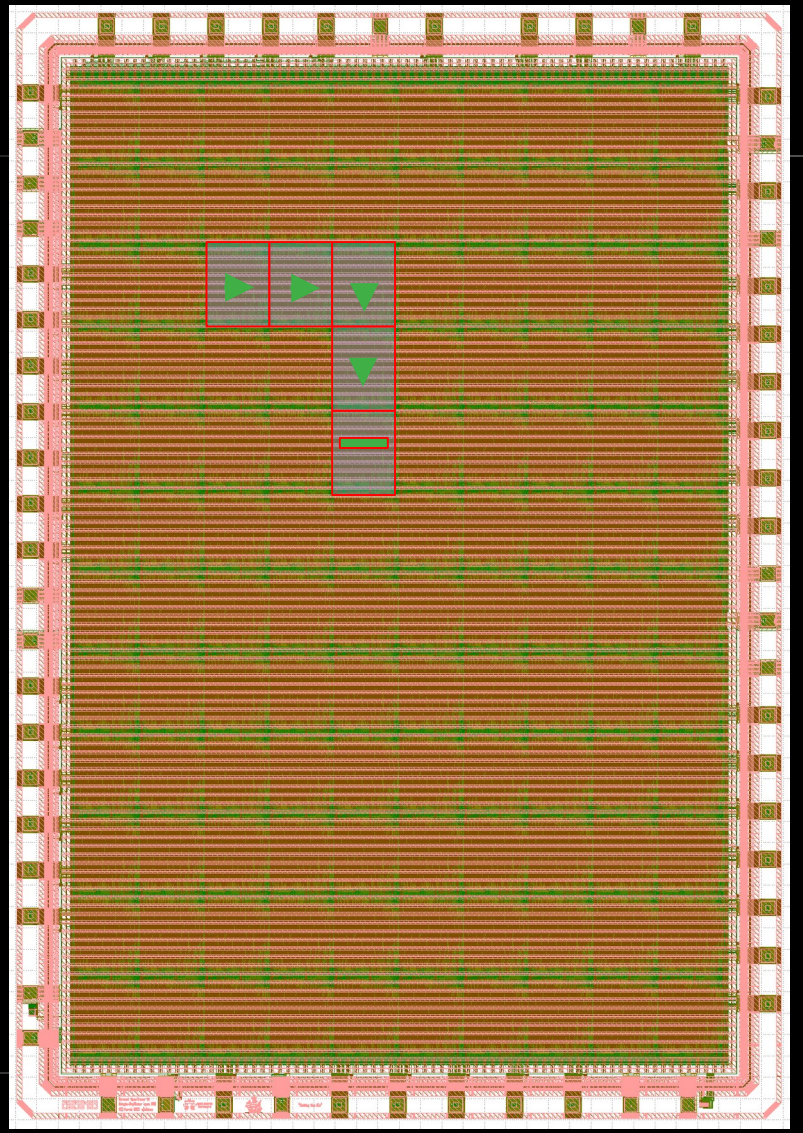
The cell

- Each cell is a small async MISC machine
  - 16 bit instruction and data, 13 instructions
  - 32 words of memory with 0 address as accumulator
  - 16x16 MAC unit, adder, logic functions
  - Executes instructions in its internal memory
- When a cell is executing, it blocks passing data and receiving new incoming data
- Communicates with 4 neighbors
  - Receive data on all 4 faces
  - Transmit data on any one configurable face

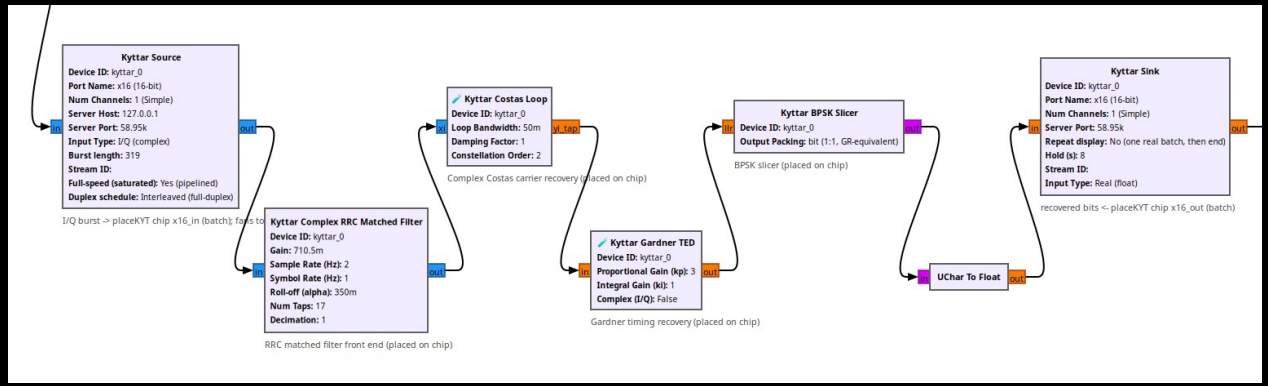


The cell

- Each cell is a small async MISC machine
  - 16 bit instruction and data, 13 instructions
  - 32 words of memory with 0 address as accumulator
  - 16x16 MAC unit, adder, logic functions
  - Executes instructions in its internal memory
- When a cell is executing, it blocks passing data and receiving new incoming data
- Communicates with 4 neighbors
  - Receive data on all 4 faces
  - Transmit data on any one configurable face
- JUMP/WRITE instructions initiate execution and write data to registers in other registers using relative addressing
  - Pass through other cells, incrementing the address until value matches the desired cell



How does this map to GNU Radio?



- Special source/sink blocks define the execution boundary
- Blocks that don’t fit in one cell span multiple cells
- ‘Compilation’ is spatially mapping blocks and their connections to the array
- Changing block functionality is a register write from the input port, through the array, to the desired cell’s register, while that part is running

Using placeKYT

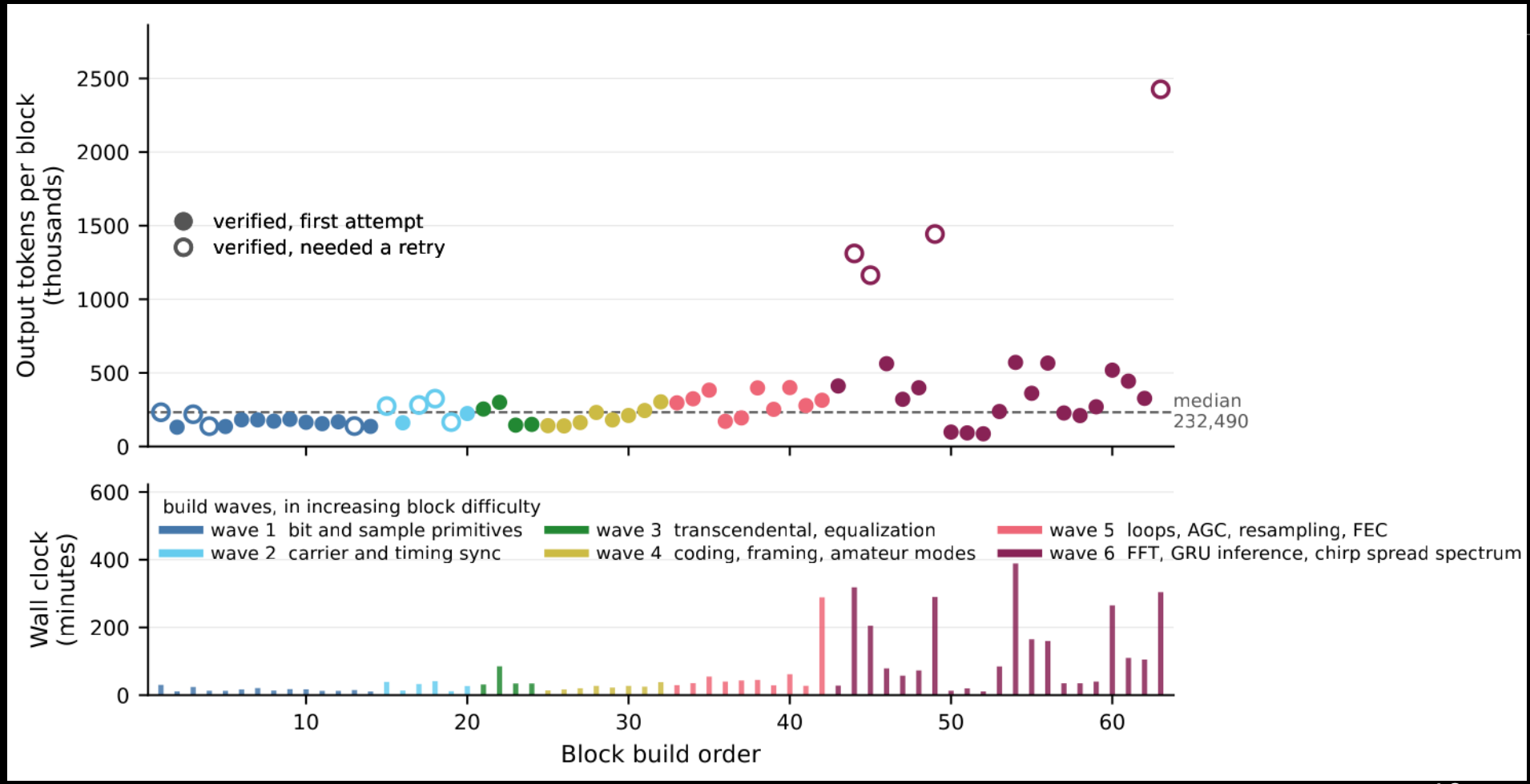
- Build your application in GNU Radio Companion using the Kyttar block library
- Import the GRC flow graph
  - Placement and routing take place automatically
- Start the server in placeKYT
- Run the simulation in GRC

Building blocks with AI

- All 113 blocks built so far are using agentic AI:
  - The ISA and cell are very simple, models can host all the information about them in their context
  - Each block constructed records lessons learned in a markdown knowledge base
  - Every block is qualified against a known good GNU Radio equivalent
- AI isn’t inventing, it is translating

Category	Representative blocks
Arithmetic and type	add, subtract, multiply, gain, constants (real and complex); float/char and complex/float conversion; conjugate, magnitude, magnitude-squared, argument, absolute value, $n \log_{10}$
FIR and IIR filters	low, high, band-pass, band-reject; root-raised-cosine matched and pulse-shaping; IIR biquad; moving average; DC blocker; complex/IQ variants of each
Mixing and frequency	complex mixer, NCO, IQ upconversion, frequency-translating FIR, frequency modulator
Rate conversion	upsampler, decimating and interpolating FIR, rational resampler, repeat, keep-one-in-N
Synchronization and equalization	Costas loop, 16-QAM constellation receiver, band-edge FLL, Mueller-and-Müller timing recovery, 4-FSK sync-word timing recovery, LMS linear equalizer
Symbol mapping and slicing	PSK, 4-FSK, and 16-QAM mappers and slicers; binary slicer; soft-decision demodulator
Coding and framing	Hamming, Golay, CRC-16, block interleaver, differential encode/decode, LFSR scrambler, pack/unpack k bits, Varicode
Conditioning	AGC (real and complex), squelch, RMS
Amateur-mode native	CW keyer and decoder, raised-cosine envelope
Transforms	streaming radix-2 FFT at 16, 32, 64, and 128 points; butterfly; twiddle multiply; multi-cell complex delay line
Machine learning	gated recurrent unit (GRU) cell with classifier head; sigmoid; tanh; dot-product multiply-accumulate
Chirp spread spectrum	chirp generator, symbol mapper, conjugate-chirp dechirp mixer, preamble sync, bin argmax
Logic and routing	XOR, NOT, AND-const, map, delay, stream splitter, crossover

Block Generation Results



Applications

- 9 complete modems/transcievers
  - BPSK, QPSK, 16QAM, 4FSK, SSB, FM, AM, CSS (receiver), PSK31, and CW
- GRU 4 signal classifier chain
- 12 miscellaneous examples using the constructed blocks

Conclusion

- For automated AI translation of GNU Radio DSP blocks to a new ISA on an MPPA, it works!
- Use the best possible model you can afford
- Opening up lessons learned, knowledge base, and block library for AI consumption will make future LLMs even better